1993-03

Improvement of miss distance of missiles with imaging seekers by utilizing dynamic image processing.

Francisco, Rui Manuel Alves

Monterey, California: Naval Postgraduate School

UNCLASSIFIED

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable): EC	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Improvement of Miss Distance of Missiles with Imaging Seekers by Utilizing Dynamic Image Processing

12. PERSONAL AUTHOR(S)
Rui Manuel Alves Francisco

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Engineer's Thesis	FROM 08/92 TO 03/93	March 1993	158

16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Tactical Missile Guidance, Miss Distance, Optimal Guidance, Dynamic Image Processing, Missile/Target 3-D Engagement, Imaging Seekers

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis deals with improving the miss distance of a missile, with imaging seeker(s), by utilizing dynamic image processing. In an encounter with a missile, a target tries to avoid the missile by performing a evasive maneuver when the missile is at a relative distance which maximizes the miss distance. Dynamic image processing permits us to identify the evasive maneuver of the target by estimating its acceleration in magnitude and direction. This thesis studies methods of utilizing this additional information about the target's behavior in order to improve the missile's performance. First the proportional navigation guidance law is explored in order to verify its advantages and weaknesses. Then, methods of obtaining the time dependent 3-D movement of a target from its image plane feature point correspondences are derived. The 3-D components of the target's acceleration are obtained by using a Kalman filter. Missiles with two cameras, one camera and one seeker (radar or IR), and only one camera are considered. Methods to get stereo vision by using the one camera plus one seeker setup and the single camera setup are proposed. Advanced guidance laws, namely advanced proportional navigation and optimal guidance, are derived for a 3-D environment. A three dimensional simulation program is developed using classical proportional navigation, advanced proportional navigation, and optimal guidance. The engagement is simulated using state variable design and the performance of the guidance laws is compared.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL: Jeffrey B. Burl	22b. TELEPHONE (Include Area Code): (408) 656-2390	22c. OFFICE SYMBOL: EC/Bl

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

Approved for public release; distribution is unlimited

Improvement of Miss Distance of Missiles with Imaging Seekers by Utilizing Dynamic Image Processing

by
Rui Manuel Alves Francisco
Lieutenant, Portuguese Navy
B.S, Portuguese Naval Academy, 1985

Submitted in partial fulfillment of the
requirements for the degrees of

ELECTRICAL ENGINEER and MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March, 1993

ABSTRACT

This thesis deals with improving the miss distance of a missile, with imaging seeker(s), by utilizing dynamic image processing. In an encounter with a missile, a target tries to avoid the missile by performing an evasive maneuver when the missile is at a relative distance which maximizes the miss distance. Dynamic image processing permits us to identify the evasive maneuver of the target by estimating its acceleration in magnitude and direction. This thesis studies methods of utilizing this additional information about the target's behavior in order to improve the missile's performance. First the proportional navigation guidance law is explored in order to verify its advantages and weaknesses. Then, methods of obtaining the time dependent 3-D movement of a target from its image plane feature point correspondences are derived. The 3-D components of the target's acceleration are obtained by using a Kalman filter. Missiles with two cameras, one camera and one seeker (radar or IR), and only one camera are considered. Methods to get stereo vision by using the one camera plus one seeker setup and the single camera setup are proposed. Advanced guidance laws, namely advanced proportional navigation and optimal guidance are derived, for a 3-D environment. A three dimensional simulation program is developed using classical proportional navigation, advanced proportional navigation, and optimal guidance. The engagement is simulated using state variable design and the performance of the guidance laws is compared.

TABLE OF CONTENTS

to my late father...
Jose Antonio Francisco
and to my mother
Maria Alves Roleira Francisco

ACKNOWLEDGEMENTS

I would like to express my gratitude to Professor J. B. Burl for his assistance, encouragement, ideas and suggestions.

I would like to express my appreciation to my wife M. Adelaide and my daughter A. Filipa for their support, encouragement and understanding.

I. INTRODUCTION

The U.S, as a result of the highly effective kamikaze attacks during World War II on U.S vessels, initiated the development of the first tactical missile (Lark guided missile). Since that time, proportional navigation guidance has been used in virtually all the world's endoatmospheric tactical radar, infrared(IR), and television(TV) guided missiles. Proportional guidance works well not only for predictable targets, but also for highly responsive ones (i.e. targets executing evasive maneuvers). The proportional navigation guidance technology currently in use appears to be adequate, if the effective time constant of the guidance system is short in comparison with the flight time and, if the missile has considerable acceleration advantage over the target. The popularity of this interceptor guidance law is the result of its simplicity of implementation, and effectiveness. Although proportional navigation was apparently known by the Germans during World War II, no applications of it were reported. In the U.S, this guidance law was studied under the auspicious of the U.S Navy. Proportional navigation was originally conceived from physical reasoning. The mathematical derivation of the "optimatility" of proportional navigation came more than 20 years later.

This research develops a three dimensional missile/target simulation using three techniques of interceptor guidance, namely classical proportional navigation, augmented proportional navigation and optimal guidance. The primary research goal is to improve the miss distance of a missile with imaging seeker(s) by utilizing dynamic image processing. The existent dynamic image processing algorithms can be used to estimate motion parameters of the target. This additional information, about the target behavior, will be included in the proportional navigation homing loop in order to increase the missile percentage of kill by improving the final miss distance. Information about the target motion is especially important in the final phase of the engagement, given that an evasive maneuver performed by the target creates appreciable miss distance that may preclude a target kill. In an encounter with a missile, a target tries to avoid the missile by performing

a evasive maneuver when the missile is at a relative distance that maximizes the miss distance. A simulation of the adjoint model of the linearized homing loop permits us to obtain miss distance projections as a function of flight time or, if preferable, as a function of the time to go. The target can induce the most miss distance by executing an evasive maneuver at a short time to go. More precisely, the optimal evasion from the target "point of view" would be a series of maneuvers at the times of flight that, by superposition, produce the most miss distance. Estimating the target maneuver and incorporating this information into the guidance control input is perhaps the difference between success and failure.

Chapter II introduces the idea of proportional navigation and how the actual guidance law is developed. Chapter III deals with estimating the target motion parameters by using two perspective views. In Chapter IV, the augmented proportional navigation and optimal guidance will be derived. Also in this chapter, a tridimensional missile/target simulation is developed using classical proportional navigation. Subsequently, the target's estimated motion parameters will be incorporated in the tridimensional engagement by using augmented proportional guidance and optimal guidance. Chapter V consists of actual simulation results. The different control laws will be tested and compared by producing miss distance projections for different evasive maneuvers. Finally, conclusions and recommendations follow in Chapter VI. All computer simulations are developed using Matrix Laboratory(MATLAB).

II. FUNDAMENTALS OF TACTICAL MISSILE GUIDANCE

A. GENERAL

Proportional navigation guidance (PROPNAV) commands the missile to turn at a rate proportional to both the angular velocity of the line of sight (LOS) and the closing velocity. The constant of proportionality is a unitless designer chosen gain (usually in the range 3-5) known as the effective navigation ratio/constant. Mathematically, the guidance law can be stated as

$$u_m = NV_c\dot{\lambda}, \qquad \text{(Eq 2.1)}$$

where u_m is the acceleration command which is perpendicular to the instantaneous LOS. N is the effective navigation constant V_c is the closing velocity along the LOS, and λ is the LOS angle (in rad). The overdot indicates the time derivative.

If the navigation ratio is greater than 1, the missile will be turning faster than the LOS, and thus the missile will build up a lead angle with respect to the line of sight. For a constant velocity missile and target the generation of this lead angle can put the missile on a collision course with the target (zero angular velocity of the line of sight). If $N = 1$ then the missile is turning at the same rate as the LOS, or simply homing on the target. If $N < 1$, then the missile will be turning slower than the LOS, thus continually falling behind the target, making an intercept impossible. In order to completely understand the physics of proportional navigation guidance it is necessary to analyze pursuit and constant bearing guidance.

B. PURSUIT GUIDANCE

For pursuit guidance, the missile velocity vector is always directed toward the target as illustrated by Figure 2.1. The missile is then constantly heading along the line of sight from the missile to the target and its path describes a pursuit path. Given that the rate of turn of the missile is always equal to the rate of turn of the LOS, "pure" pursuit (without leading angle) paths are highly curved. This requires the missile to use significant acceleration.

Since the signal processing is limited to continuously locating the target and changing the missile flight path angle, the on-board avionics are relatively simple. As will be demonstrated later, this kind of classical guidance law is a special case of PROPNAV when the effective navigation ratio is equal to 1.

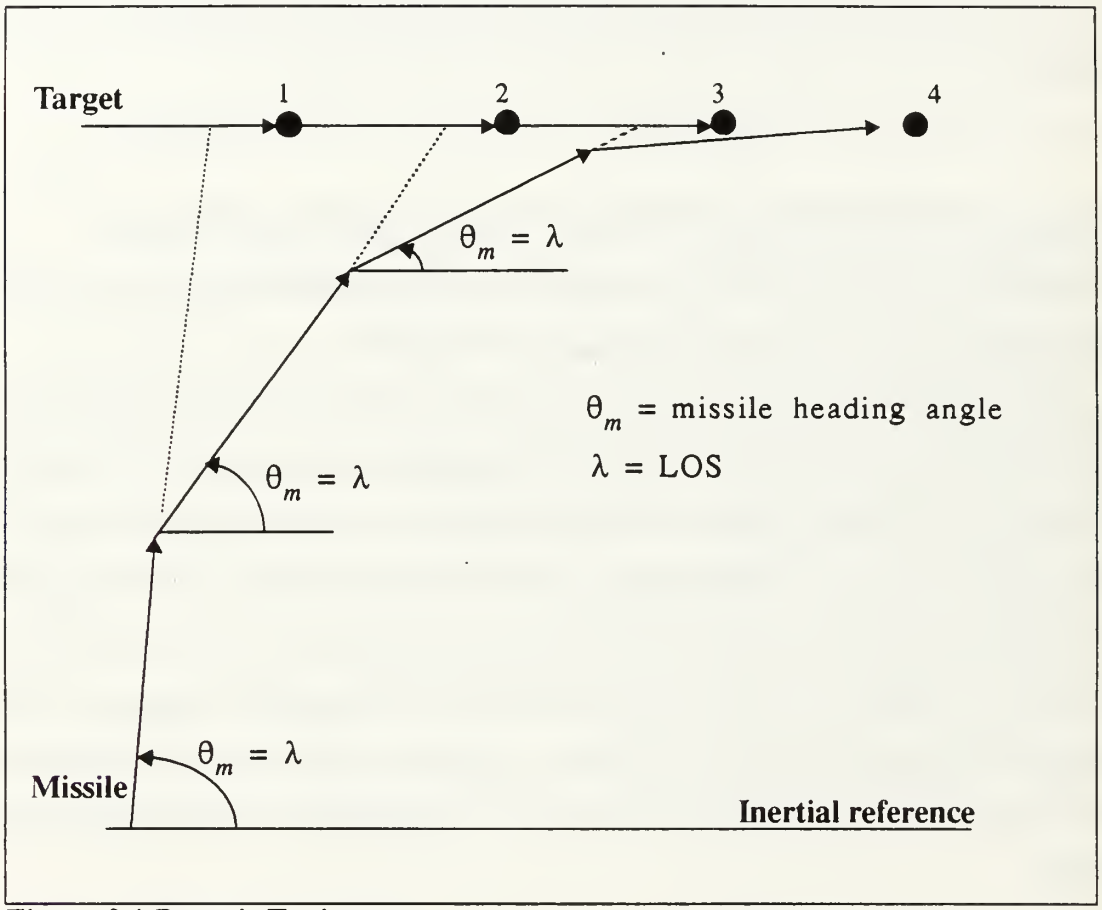

Figure 2.1 Pursuit Trajectory

Figure 2.2 shows the geometry of the pursuit guidance law. V_m and V_t are respectively the missile and target velocities, θ_m and θ_t are respectively the missile and target flight path angles, α_t is the difference between the LOS angle and the target flight path angle, and

r is the instantaneous separation between missile and target. Inertial and missile translating coordinate systems are also shown in the figure.

The velocity of the target with respect to the missile is given by:

$$\vec{v} = \vec{V}_t - \vec{V}_m; \qquad \text{(Eq 2.2)}$$

$$\vec{v} = \dot{r}\vec{e}_r + r\dot{\theta}\vec{e}_\theta. \qquad \text{(Eq 2.3)}$$

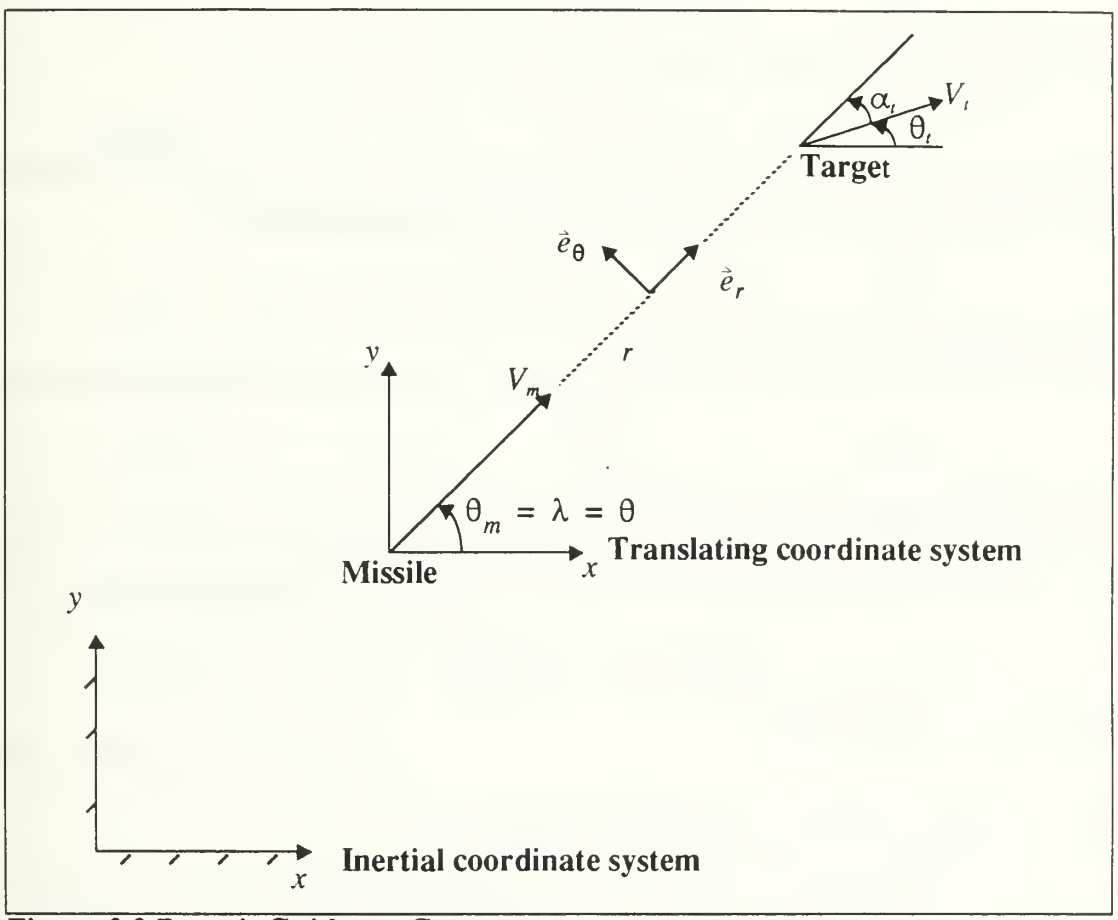

Figure 2.2 Pursuit Guidance Geometry

Writing the velocity of the target and missile in terms of the polar unit base vectors $\vec{e}_r$ and $\vec{e}_\theta$, we get:

$$\vec{V}_t = V_t \cos\alpha_t \vec{e}_r - V_t \sin\alpha_t \vec{e}_\theta, \quad \text{(Eq 2.4)}$$

$$\vec{V}_m = V_m \vec{e}_r. \quad \text{(Eq 2.5)}$$

Equating equations 2.2 and 2.3 and using equations 2.4 and 2.5, we obtain:

$$\dot{r} = V_t \cos\alpha_t - V_m; \quad \text{(Eq 2.6)}$$

$$r\dot{\theta}_m = -V_m \sin\alpha_t. \quad \text{(Eq 2.7)}$$

From Figure 2.2, we see that:

$$\theta_m = \alpha_t + \theta_t. \quad \text{(Eq 2.8)}$$

Considering a non responsive target:

$$\dot{\theta}_m = \dot{\alpha}_t. \quad \text{(Eq 2.9)}$$

The missile acceleration $\vec{u}$ is obtained by differentiating equation 2.5:

$$\vec{u} = \dot{V}_m \vec{e}_r + V_m \dot{\theta}_m \vec{e}_\theta, \quad \text{(Eq 2.10)}$$

given that from analytical mechanics:

$$\frac{d\vec{e}_r}{dt} = \vec{e}_\theta \frac{d\theta}{dt}. \quad \text{(Eq 2.11)}$$

Assuming constant speed (magnitude of the velocity vector), the acceleration command will be the normal component of the acceleration which will be designated u_m:

$$u_m = V_m \dot{\theta}_m = V_m \dot{\lambda} = V_m \dot{\alpha}_t, \quad \text{(Eq 2.12)}$$

where $\dot{\alpha}_t$ is a time function.

C. CONSTANT BEARING GUIDANCE

The accelerations required by the pursuit guidance law can be reduced by aiming the missile ahead of the target by using a lead angle. In this case the missile traverses a straight line to a collision with a constant speed non maneuvering target, as shown in Figure 2.3. The missile converges on the target by using a constant LOS angle (λ = constant). Since the rate of change of the LOS angle is zero throughout the flight, the lateral accelerations are

zero. If the target maneuvers evasively, by changing its velocity vector in direction and/or in magnitude, a new collision course must be computed and the missile flight path altered accordingly. The constant bearing geometry is shown in Figure 2.4.

We wish to find the missile control input necessary to responde to prescribed target accelerations. The relative velocity is given by:

$$\vec{v} = \vec{V}_t - \vec{V}_m = \dot{r}\vec{e}_r + r\dot{\lambda}\vec{e}_\theta. \qquad \text{(Eq 2.13)}$$

The target and missile absolute velocities are:

$$\vec{V}_t = V_t\cos\alpha_t\vec{e}_r - V_t\sin\alpha_t\vec{e}_\theta; \qquad \text{(Eq 2.14)}$$

$$\vec{V}_m = V_m\cos\alpha_m\vec{e}_r - V_m\sin\alpha_m\vec{e}_\theta. \qquad \text{(Eq 2.15)}$$

Subtracting equation 2.14 from equation 2.15 and equating the result to 2.13, we find that:

$$\dot{r} = V_t\cos\alpha_t - V_m\cos\alpha_m; \qquad \text{(Eq 2.16)}$$

$$r\dot{\lambda} = -V_t\sin\alpha_t + V_m\sin\alpha_m. \qquad \text{(Eq 2.17)}$$

The requirements for a constant bearing guidance are:

$$\dot{\lambda} = 0; \qquad \text{(Eq 2.18)}$$

$$\dot{r} < 0. \qquad \text{(Eq 2.19)}$$

Using equations 2.17 and 2.18, we get:

$$\sin\alpha_m = \frac{\sin\alpha_t}{V_m}V_t, \qquad \text{(Eq 2.20)}$$

From equations 2.16 and 2.19:

$$\cos\alpha_m > \frac{\cos\alpha_t}{V_m}V_t. \qquad \text{(Eq 2.21)}$$

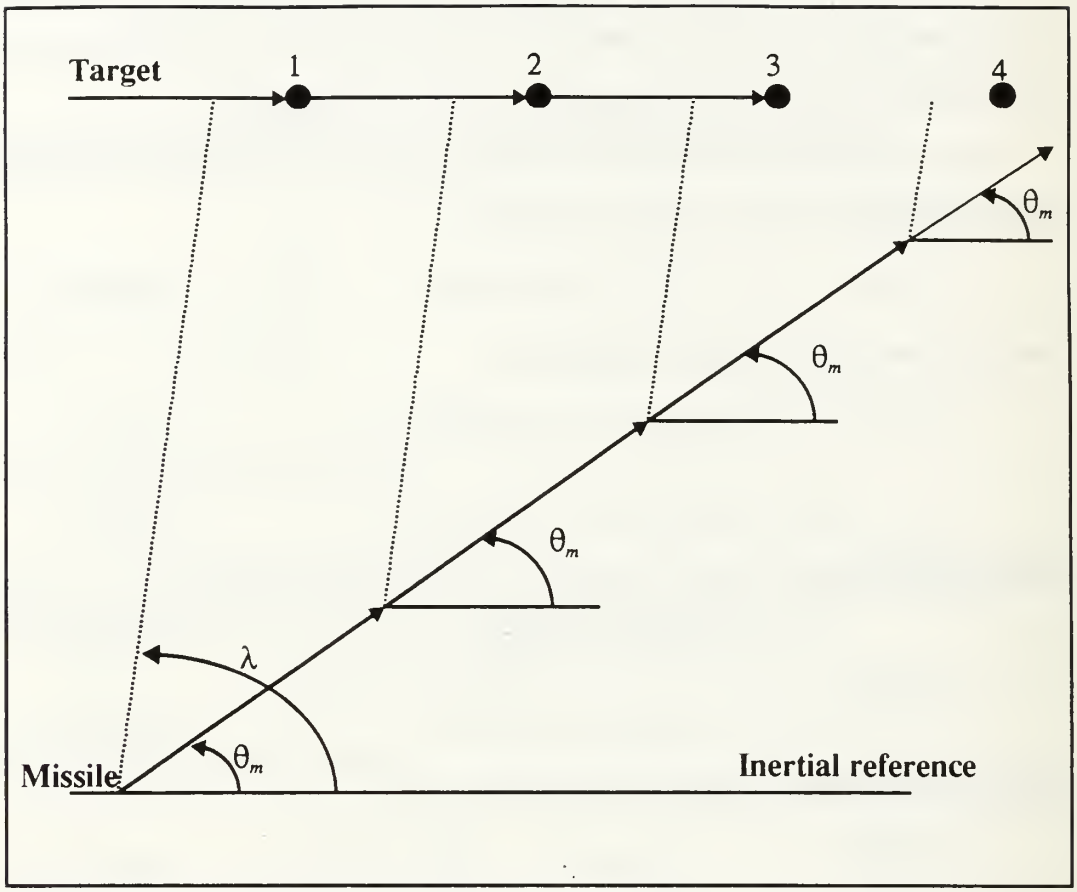

Figure 2.3 Constant Bearing Guidance Trajectory

We may use equation 2.20 to obtain a expression for $\cos\alpha_t$ by squaring the equation and using the fundamental trigonometric entity. Doing this, we get:

$$\frac{\cos\alpha_t}{V_m}V_t = \left[\left(\frac{V_t^2}{V_m^2} - 1\right) + (\cos\alpha_m)^2\right]^{\frac{1}{2}}. \qquad \text{(Eq 2.22)}$$

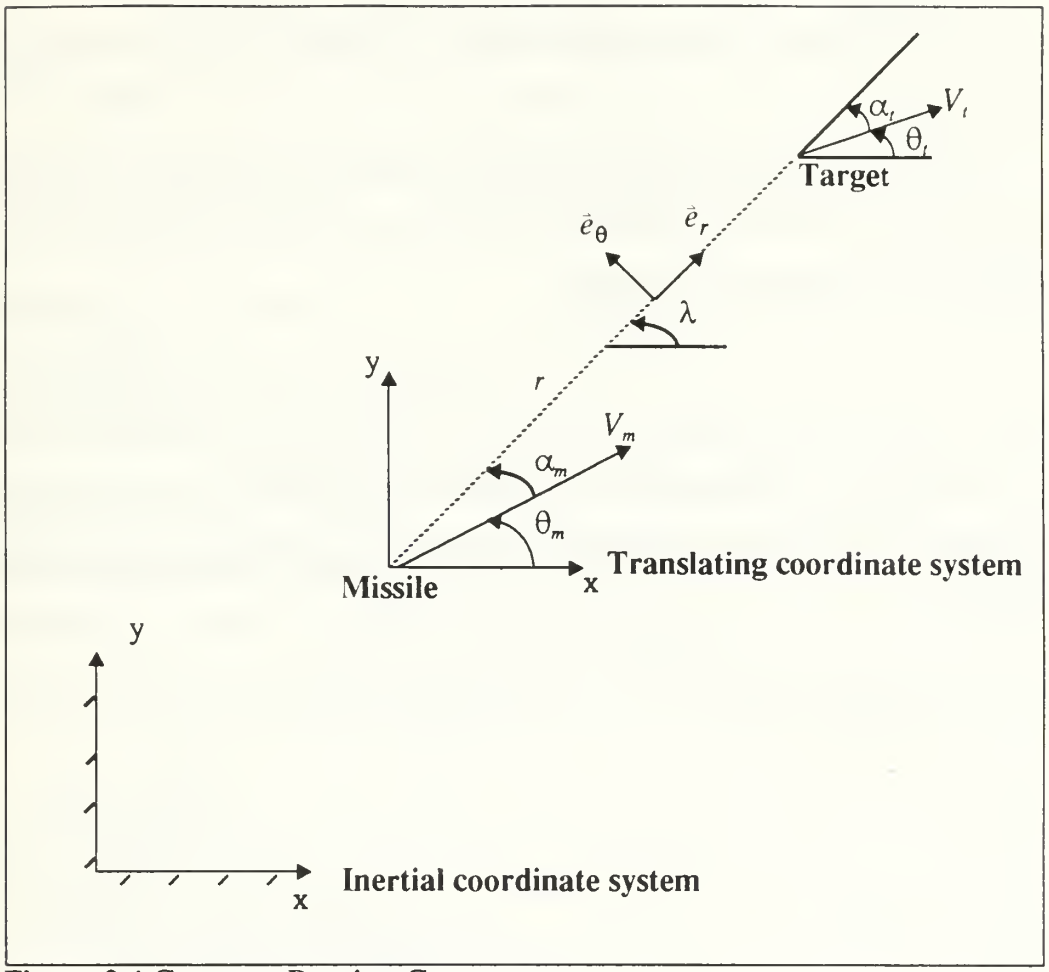

Figure 2.4 Constant Bearing Geometry

Substituting this expression into equation 2.21, we obtain:

$$\cos\alpha_m > \left[\left(\frac{V_t^2}{V_m^2} - 1 \right) + (\cos\alpha_m)^2 \right]^{\frac{1}{2}}. \qquad \text{(Eq 2.23)}$$

This expression is satisfied if:

$$\frac{V_t^2}{V_m^2} - 1 < 0 \Rightarrow V_m > V_t. \qquad \text{(Eq 2.24)}$$

Thus, for this guidance law to be effective the missile must have speed advantage relative to the target.

The missile and target accelerations can be computed from Figure 2.5, which expands the acceleration in terms of tangential and normal components. The velocity vector of a body (missile or target) is described by:

$$\vec{V} = V\hat{\tau}, \qquad \text{(Eq 2.25)}$$

where $\hat{\tau}$ represents the tangential unit vector to the trajectories represented by the dashed lines in the figure. The figure shows the response of the missile to a evasive target. We are interested in computing the relationship between the missile control input and the target maneuver.

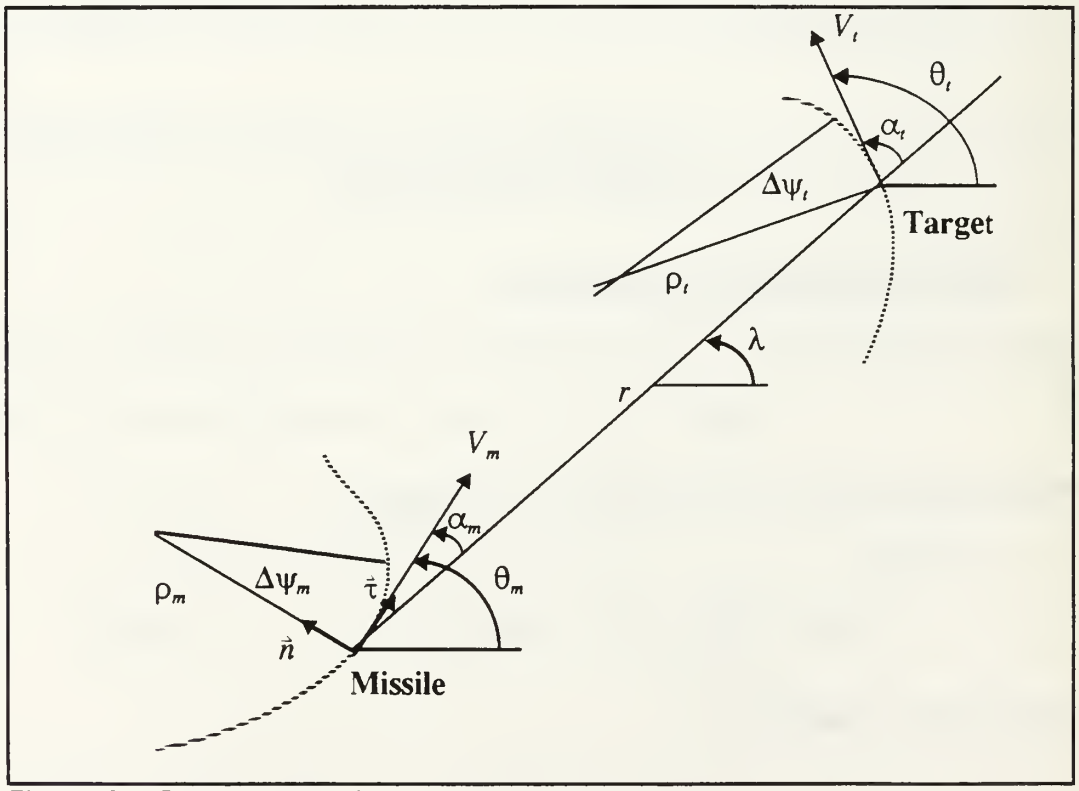

Figure 2.5 Constant Bearing Normal Acceleration

The acceleration of the body is given by:

$$\vec{u} = \frac{d\vec{V}}{dt} = \frac{d}{dt}(V\vec{\tau}) = \dot{V}\vec{\tau} + V\frac{d\vec{\tau}}{dt}. \qquad \text{(Eq 2.26)}$$

For small values of $\Delta\psi$ it approaches the magnitude of $\Delta\vec{\tau}$ and the direction of $\Delta\vec{\tau}$ becomes perpendicular to the direction of $\vec{\tau}$. It follows that the derivative $\frac{d\vec{\tau}}{d\vec{\psi}}$ is of magnitude 1 and perpendicular to $\vec{\tau}$. Then this derivative is the unit normal vector $\vec{n}$. The time derivative $\frac{d\vec{\tau}}{dt}$ is found by using the chain rule, as follows:

$$\frac{d\vec{\tau}}{dt} = \frac{d\vec{\tau}}{d\vec{\psi}}\frac{d\vec{\psi}}{dt} = \vec{n}\dot{\psi} = \vec{n}\dot{\theta}. \qquad \text{(Eq 2.27)}$$

Assuming constant speed, the missile and target accelerations are always in the direction of the respective unit normal components and can be written as:

$$\vec{u}_m = \vec{n}_m V_m \dot{\theta}_m; \qquad \text{(Eq 2.28)}$$

$$\vec{u}_t = \vec{n}_t V_t \dot{\theta}_t \qquad \text{(Eq 2.29)}$$

since,

$$\theta_m = \alpha_m + \lambda, \qquad \text{(Eq 2.30)}$$

and given that $\dot{\lambda} = 0$,

$$\dot{\theta}_m = \dot{\alpha}_m, \qquad \text{(Eq 2.31)}$$

$$u_m = V_m \dot{\alpha}_m. \qquad \text{(Eq 2.32)}$$

The variable, u_m is the missile acceleration magnitude. Differentiating equation 2.20, we find $\dot{\alpha}_m$ and then u_m in terms of the target evasive acceleration:

$$\dot{\alpha}_m(t) = \left(\frac{\cos\alpha_t(t)}{V_m\cos\alpha_m(t)}V_t\right)\dot{\alpha}_t(t), \qquad \text{(Eq 2.33)}$$

where the time factor is included. Additionally,

$$\dot{\alpha}_t(t) = \dot{\theta}_t(t) = \frac{u_t(t)}{V_t} \qquad \text{(Eq 2.34)}$$

where $u_t(t)$ is the target acceleration magnitude. Hence:

$$u_m(t) = \left(\frac{\cos\alpha_t(t)}{\cos\alpha_m(t)}\right) u_t(t) . \qquad \text{(Eq 2.35)}$$

From this last equation and equation 2.20, we conclude that the LOS will maintain its direction in space, keeping the missile on a collision course with the target provided that the missile's and target's kinematics normal to the LOS behave likewise. Additionally, from equation 2.21, the closing velocity (component of the relative velocity along the LOS) must be positive. Constant bearing guidance requires the knowledge of the heading and velocity of the target, the line of sight, and the velocity of the missile, which dictates a more complex signal processing system than for pursuit guidance.

D. PROPORTIONAL NAVIGATION GUIDANCE

1. In Search Of The Proportional Navigation Concept

Pursuit guidance tries to continuously point the missile to the target, resulting a highly curved path and very large accelerations. The guidance law is only interested in the present position of the target; lacking information about the target kinematics. This lack of information precludes the missile from building a lead angle, resulting in a somewhat ineffective guidance law. Constant bearing guidance points the missile to the future position of the target, resulting in a straight line collision path with a non maneuvering target. Before pointing the missile, the guidance system needs to know the heading and velocity of the target to compute the target's future position. So, this method is not practical, especially when dealing with targets with evasive capabilities.

The advantage of proportional navigation is that it provides a practical method of approximating a constant bearing course to a maneuvering target. PROPNAV tries to emulate the constant bearing guidance command by using LOS rate information from an on - board electromagnetic or electro - optic device.

A missile using constant bearing guidance only needs a control input when it is necessary to change its heading at the beginning of the flight and afterwards if the target maneuvers. The form of this command signal was derived and is repeated here for convenience:

$$\vec{u}_m = V_m \dot{\alpha}_m \vec{n}. \qquad \text{(Eq 2.36)}$$

From the proportional navigation geometry in Figure 2.6:

$$\theta_m = \alpha_m + \lambda. \qquad \text{(Eq 2.37)}$$

Taking its derivative:

$$\dot{\theta}_m = \dot{\alpha}_m + \dot{\lambda}. \qquad \text{(Eq 2.38)}$$

The acceleration of the missile (assuming constant speed) is:

$$\vec{u}_m = V_m \dot{\theta}_m \vec{n} = V_m (\dot{\alpha}_m + \dot{\lambda}) \vec{n},. \qquad \text{(Eq 2.39)}$$

Our goal is to emulate equation 2.36 by using a linear transformation between the LOS rate $\dot{\lambda}$ and the missile's angle rate $\dot{\alpha}_m$. Set, for example:

$$\dot{\lambda} = \frac{1}{N-1} \dot{\alpha}_m. \qquad \text{(Eq 2.40)}$$

Equation 2.39 becomes:

$$\vec{u}_m = V_m (\dot{\alpha}_m + \frac{1}{N-1} \dot{\alpha}_m) \vec{n} = V_m \frac{N}{N-1} \dot{\alpha}_m \vec{n}. \qquad \text{(Eq 2.41)}$$

By letting N be large this equation approaches equation 2.36 for a constant bearing path (collision course). We are interested in the relationship between the LOS rate and the flight path angle rate. Using equations 2 38 and 2.40, we find that:

$$\dot{\theta}_m = \dot{\alpha}_m + \dot{\lambda} = (N-1)\dot{\lambda} + \dot{\lambda} = N\dot{\lambda}; \qquad \text{(Eq 2.42)}$$

$$\vec{u}_m = N V_m \dot{\lambda} \vec{n}. \qquad \text{(Eq 2.43)}$$

Therefore, PROPNAV is a practical guidance law that emulates constant bearing guidance by issuing control commands that are proportional to the LOS rate.

Pursuit guidance is a particular case of proportional navigation when N =1 (compare equations 2.12 and 2.43). As we have seen, constant bearing guidance is obtained by letting N be large (theoretically infinity). However, large gains in the amplifiers also cause large amplifications of noise; therefore N is usually restricted to less than 6. proportional navigation paths are less curved than pursuit paths, but more curved than constant bearing collisions. PROPNAV anticipates the future position of the target without actually computing it. Due to this property this guidance law presents a higher degree of responsiveness than other guidance laws.

2. Proportional Navigation And Zero Effort Miss

In Figure 2.6 the missile, with velocity magnitude V_m, is heading at an angle of $L + HE$ with respect to the line of sight. The angle L is known as the missile lead angle and is the theoretically correct angle for the missile to be on a collision triangle, with the target. If the missile is launched in a collision triangle with a non evasive target, no further accelerations commands will be required to hit the target. The angle HE is known as the heading error, and represents the initial deviation of the missile from the collision triangle.

In practice, the missile is usually not launched exactly in a collision triangle, since the expected intercept point is not known precisely. The location of the intercept point can only be approximated, because we do not know in advance what the target will do in the future. In fact, that is why a guidance system is required. The point of closest approach of the missile and target is known as the miss distance. Guidance system lags or subsystem dynamics will cause miss distance. The simplest proportional navigation homing loop is shown in Figure 2.7 where we have linearized the missile/target engagement by using the small angles approximation (i.e. we assume that the flight-path angles and the line of sight angle are small in order to linearize the engagement geometry. Then, the cosine functions are approximated by 1 and the sine and tangent functions by their arguments). In a linearized analysis, the range equation is approximated by the following time varying relationship:

$$r = V_c(t_F - t) = V_c t_{go} \qquad \text{(Eq 2.44)}$$

where V_c is the closing velocity, t_F is the total flight time, and t_{go} (time to go) is the time until the end of the flight. As shown in [Ref. 1] the miss distance will always be zero in a zero-lag proportional navigation homing loop. The PROPNAV guidance law used in the homing loop of Figure 2.7, and also the most used in the literature, is not the one derived in equation 2.43, but the following one:

$$\vec{u}_m = NV_c\dot{\lambda}\vec{n}_\lambda \qquad \text{(Eq 2.45)}$$

where $\vec{n}_\lambda$ is the unit vector normal to the LOS. Then, the control input is issued perpendicular to the instantaneous LOS. It can be easily demonstrated that this last expression maintains the proportionality between the missile flight path angle and the angular LOS rate. In [Ref. 2], Guelman contrasted "pure" PROPNAV (described by equation 2.43, wherein command accelerations are normal to the missile velocity vector) and "true" PROPNAV (described by equation 2.45, wherein command accelerations are normal to the line of sight). He concluded that the later law would result in intercept only if the initial conditions were within a well-defined subset of the parameter space. In the homing loop of Figure 2.7, the seeker provides the LOS rate by taking the derivative of the geometric LOS angle. The noise filter processes the noisy LOS rate measurements to provide an estimate of the LOS rate. The guidance command is generated using the "true" PROPNAV guidance law. The guidance system must cause the missile to maneuver, by using moving control surfaces. The seeker and the guidance system dynamics are described by differential equations.

The presence of delays in the homing loop creates miss distance. In the presence of guidance system dynamics, the heading error (*HE*) and target maneuver (target evasive acceleration) are the two sources of miss distance. The PROPNAV guidance law can be expressed in terms of the zero effort miss. The zero effort miss is not only useful in explaining PROPNAV but is also useful in deriving more advanced missile control laws.

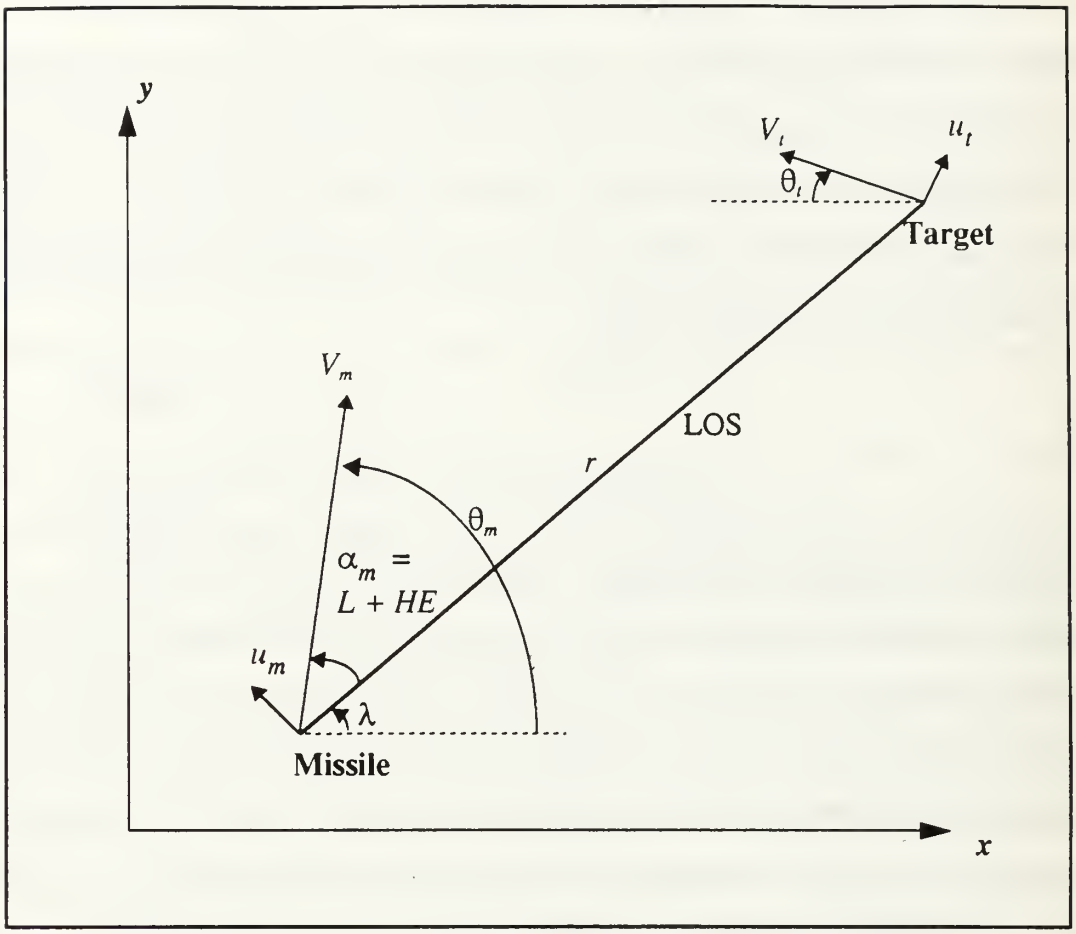

Figure 2.6 Proportional Navigation Two Dimensional Engagement

The zero effort miss is the distance the missile would miss the target if the target continued along its present course and the missile made no further corrective maneuvers. Using Figure 2.6:

$$ZEM_x = r_x + v_x t_{go}, \qquad \text{(Eq 2.46)}$$

$$ZEM_y = r_y + v_y t_{go}; \qquad \text{(Eq 2.47)}$$

where ZEM represents the zero effort miss, r is the missile/target relative distance, and v is the missile/target relative speed. The subscript *(*x or *y)* represent the projection of the respective quantity over that coordinate axis.

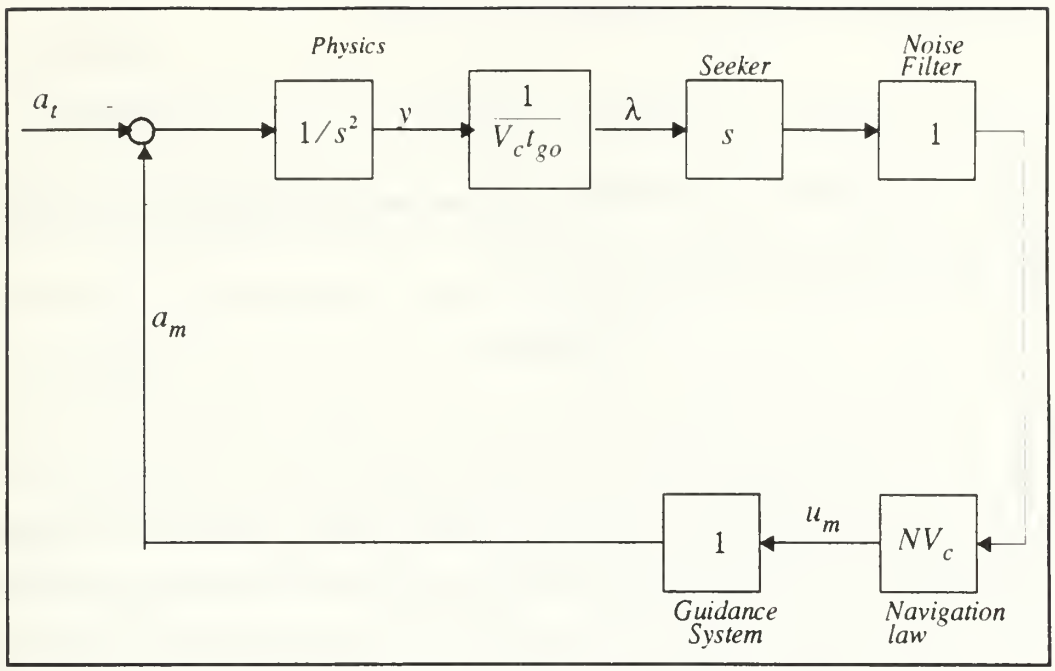

Figure 2.7 Zero - lag Proportional Navigation Homing Loop (Linearized Engagement)

The ZEM perpendicular to the LOS, is given by:

$$ZEM_{PLOS} = -ZEM_x \sin\lambda + ZEM_y \cos\lambda. \qquad \text{(Eq 2.48)}$$

Using equations 2.46 through 2.48, we obtain:

$$ZEM_{PLOS} = \frac{t_{go}(r_x v_y - r_y v_x)}{r}. \qquad \text{(Eq 2.49)}$$

The LOS is:

$$\lambda = \text{atan}\left(\frac{r_y}{r_x}\right), \qquad \text{(Eq 2.50)}$$

taking its derivative, we obtain:

$$\dot{\lambda} = \frac{r_x v_y - r_y v_x}{r^2}. \qquad \text{(Eq 2.51)}$$

Comparing equations 2.49 and 2.51, the LOS rate may be expressed in terms of the component of the zero effort miss normal to the LOS:

$$\dot{\lambda} = \frac{ZEM_{PLOS}}{rt_{go}} = \frac{ZEM_{PLOS}}{V_c t_{go}^2}; \qquad \text{(Eq 2.52)}$$

where $r = V_c t_{go}$. Then the PROPNAV guidance command magnitude can be expressed in terms of the ZEM perpendicular to the LOS:

$$u_m = \frac{N\ ZEM_{PLOS}}{t_{go}^2}. \qquad \text{(Eq 2.53)}$$

Thus, we conclude that the PROPNAV acceleration command that is perpendicular to the LOS is not only proportional to the LOS rate and closing velocity but is also proportional to the zero effort miss and inversely proportional to the square of time to go. The efficiency of PROPNAV guidance is a direct consequence of this dynamic property. This is, of course, a very powerful concept.

III. DYNAMIC IMAGE PROCESSING

A. GENERAL

A missile that uses a TV camera and a seeker (radar or IR), or instead, two TV cameras is considered. A setup with only one TV camera is also studied. The seeker and the camera, or the two cameras, can be located on the missile's nose separated by a transversal distance d. The seeker plus the single camera setup, permits the missile to emulate the stereo vision of the two cameras setup. It has the additional advantage of tracking the target at the early stages of the engagement using solely the seeker's LOS angle information. This system permits us to compute the 3-D target motion by using a two perspective views motion algorithm and the target's spatial direction and range provided by the seeker. The two cameras setup permits us to use image plane locations in two views, corresponding to a single object point at times t_1 and t_2, to determine the 3-D object (target) locations $\vec{X}_o(t_1)$ and $\vec{X}_o(t_2)$. The one camera setup also permits us to determine the motion of the target, as a function of time. This is done by using a two perspective views motion algorithm and guessing the target's physical dimensions to estimate its absolute depth. In this way, we emulate binocular vision. The estimated 3-D motion of the target and the image sampling time permit us to estimate the target velocity and acceleration components in a preselected 3-D rectangular coordinate system. The acceleration information can subsequently be injected into the control algorithms, which will be developed in the next chapter, to improve the miss distance.

1. Scene (3-D) - Image (2-D) Geometric Considerations

Mathematically, we can express the transformation of object point locations (3-D) to image plane locations (2-D) by the following generally noninvertible geometric transformation:

$$\vec{X}_i(t) = g(\vec{X}_o(t), \ldots). \qquad \text{(Eq 3.1)}$$

The modeling of the imaging process, described by the above equation, relates object points $\vec{X}_o(t)$ in the 3-D scene to image points $\vec{X}_i(t)$ in the image plane. The function g depends on the imaging geometry, lens model, and coordinate system choices.

The imaging model is derived by considering the pinhole camera model shown in Figure 3. 1. The point $\vec{X}_c$ lies over the camera's optical axis at a distance f from the image plane. The figure shows two distinct coordinate systems, an image plane coordinate system and a global coordinate system. Our first goal is: assuming the simplified camera model shown in Figure 3.1, derive the transformation described by equation 3.1 where the object point $\vec{X}_o(t)$ can be measured from either coordinate system.

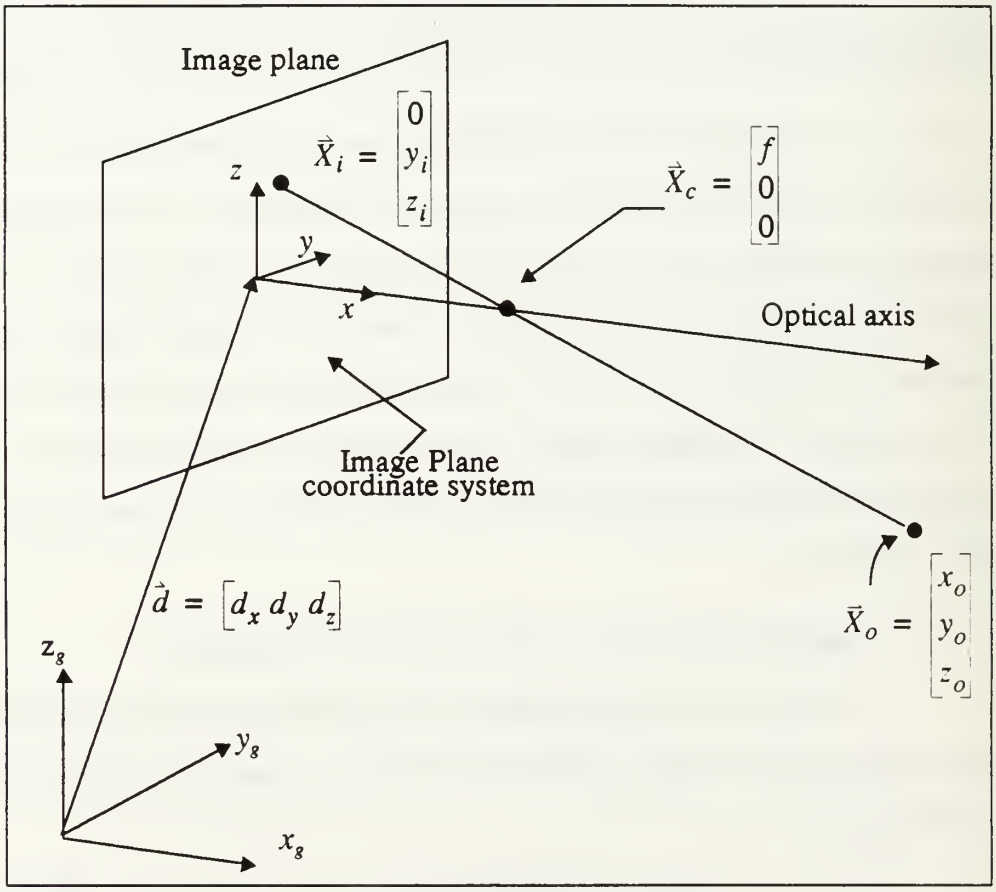

Figure 3.1 Scene - Image Transformation

The object/image relationship defined in equation 3.1 is defined by a transformation matrix. Independent of the camera model, this transformation matrix is the product of two matrices. The first matrix describes the object - image coordinates transformation, and is derived by assuming that the 3-D object coordinates are measured relatively to the image plane. However, if the object coordinates are measured relatively to the global coordinate system, a second transformation matrix relating the two coordinate systems have to be defined. This matrix is the composite of the relative rotation and translation between the coordinate systems. It describes the coordinates transformation between the two coordinate systems.

To identify the transformation defined by equation 3.1, the two matrices are derived for the simplified camera model of Figure 3.1. Monocular vision (only one camera) is incapable of determining absolute depth. However, any imaged point is constrained to correspond to an object point located anywhere on the 3-D line segment containing $\vec{X}_i(t)$, $\vec{X}_c$ and $\vec{X}_o(t)$.

Assuming that the coordinate systems for both object and image points, are coincident and centered in the image plane, the above colinear points are related by:

$$k(\vec{X}_i(t) - \vec{X}_c) = (\vec{X}_c - \vec{X}_o(t)). \qquad \text{(Eq 3.2)}$$

Expanding this equation yields:

$$k\{\begin{bmatrix}0 & y_i & z_i\end{bmatrix}^T - \begin{bmatrix}f & 0 & 0\end{bmatrix}^T\} = \begin{bmatrix}f & 0 & 0\end{bmatrix}^T - \begin{bmatrix}x_o & y_o & z_o\end{bmatrix}^T, \qquad \text{(Eq 3.3)}$$

where the superscript T is the transpose operator. The time index t has been dropped to simplify the notation. Equation 3.3 yields:

$$k = \frac{(x_o - f)}{f} = \frac{x_o}{f} - 1, \qquad \text{(Eq 3.4)}$$

$$y_i = -\frac{y_o}{k} = \frac{fy_o}{(f - x_o)}, \qquad \text{(Eq 3.5)}$$

$$z_i = -\frac{z_o}{k} = \frac{fz_o}{(f - x_o)}. \qquad \text{(Eq 3.6)}$$

The minus sign in the second expression of the two last equations, stands for the image inversion originated by the back - projection model of Figure 3.1. The matrix representation of the nonlinear equations 3.5 and 3.6 is:

$$\vec{X}_i = M\vec{X}_o = \begin{bmatrix} 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ -1 & 0 & 0 & f \end{bmatrix} \vec{X}_o. \qquad \text{(Eq 3.7)}$$

The last equation uses homogeneous coordinates (a technique also used to develop computer graphics), for image and object points. The homogeneous coordinates are defined by multiplying the physical coordinates by an arbitrary constant c and including the constant as an additional element of the vector:

$$\hat{X}_i = \begin{bmatrix} cy_i & cz_i & c \end{bmatrix}^T, \qquad \text{(Eq 3.8)}$$

and

$$\hat{X}_o = \begin{bmatrix} x_o & y_o & z_o & 1 \end{bmatrix}^T. \qquad \text{(Eq 3.9)}$$

Note that in equation 3.9 the arbitrary constant is equal to 1. The object - image point transformation is defined by equation 3.7, where it is implicitly assumed that $x_i = 0$. Rewriting equations 3.5 and 3.6, we conclude:

$$\frac{y_o}{y_i} = \frac{z_o}{z_i} = \frac{x_o - f}{-f}. \qquad \text{(Eq 3.10)}$$

Fixing the image plane coordinates y_i and z_i the above equation describes the 3-D line over which the 3 -D object is located. Therefore, while the transformation in equation 3.7 is not invertible, choice of a specific image point constrains corresponding object points to lie along a 3-D ray (shown in Figure 3.1).

If the object points are measured relatively to the global coordinate system, the matrix relating the two coordinate systems has to be computed. This transformation matrix

is the product of a succession of matrices. Individually, each of these matrices defines a rotation or translation of the image plane coordinate system relative to the global coordinate system. The succession of transformations may be of the form:

$$\hat{X}_o = (T2\ R2\ R1\ T1)\hat{X}_o^g, \qquad \text{(Eq 3.11)}$$

where $\hat{X}_o$ and $\hat{X}_o^g$ define the homogeneous object coordinates in the image plane and global coordinate systems, respectively. Here the first transformation is the translation T1 followed by the rotation R1, etc. The composite of the above transformation may be defined by:

$$H_{g \to i} = T2\ R2\ R1\ T1\ . \qquad \text{(Eq 3.12)}$$

Then, the general relationship between object points measured relatively to any user selected coordinate system and the image plane points is:

$$\hat{X}_i = MH_{g \to i}\hat{X}_o^g. \qquad \text{(Eq 3.13)}$$

For the simple case of only a translation as shown in Figure 3.1, we see that in object coordinates:

$$\vec{X}_o = \vec{X}_o^g - \left[d_x\ d_y\ d_z\right]^T. \qquad \text{(Eq 3.14)}$$

Homogeneous coordinates enable us to represent the last relationship using a translation matrix:

$$\hat{X}_o = H_{g \to i}\hat{X}_o^g = \begin{bmatrix} 1 & 0 & 0 & -d_x \\ 0 & 1 & 0 & -d_y \\ 0 & 0 & 1 & -d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{X}_o^g. \qquad \text{(Eq 3.15)}$$

2. Stereo Vision (2 Cameras)

Monocular vision disables depth perception. In fact, due to the impossibility of inverting the 3×4 matrix $Q = MH_{g \to i}$ obtained in the last section, we are constrained to the determination of image points from object points. However, we are interested in

determining the 3-D locations (measured relatively to a global coordinate system). One approach to solve this problem is to use more than one camera. One of our proposals, is to use two cameras in the missile's nose separated by a distance d emulating, in some way, the human visual system.

Initially, we assume the simplified two dimensional diagram of the stereo vision in Figure 3.2. The scene consists of a 2-D surface. As shown in the figure, a point on this surface is projected onto the two image planes (IP1 and IP2). In general the two centers of projection differ in length (f_1 and f_2). It is assumed that the user selected global coordinate system, to measure the object coordinates, is coincident and centered in the image plane IP1. The coordinate x_i is shown in the figure, the coordinate y_i is perpendicular to x_i and in the plane of the page. The object point may be determined using the two projected points, one in each camera.

The relationship between the homogeneous coordinates of the object point, measured relatively to the image plane IP1, and the homogeneous coordinates of the corresponding image point is:

$$\begin{bmatrix} cx_{i1} \\ c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{1}{f_1} & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}. \qquad \text{(Eq 3.16)}$$

The object coordinates measured from IP2 are related to the object coordinates measured from IP1 by the following relationship:

$$\vec{X}_o^{IP2} = \vec{X}_o^{IP1} + \begin{bmatrix} d \\ 0 \end{bmatrix}, \qquad \text{(Eq 3.17)}$$

or, in homogeneous coordinates:

$$\hat{X}_o^{IP2} = \begin{bmatrix} 1 & 0 & d \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \hat{X}_o^{IP1}. \qquad \text{(Eq 3.18)}$$

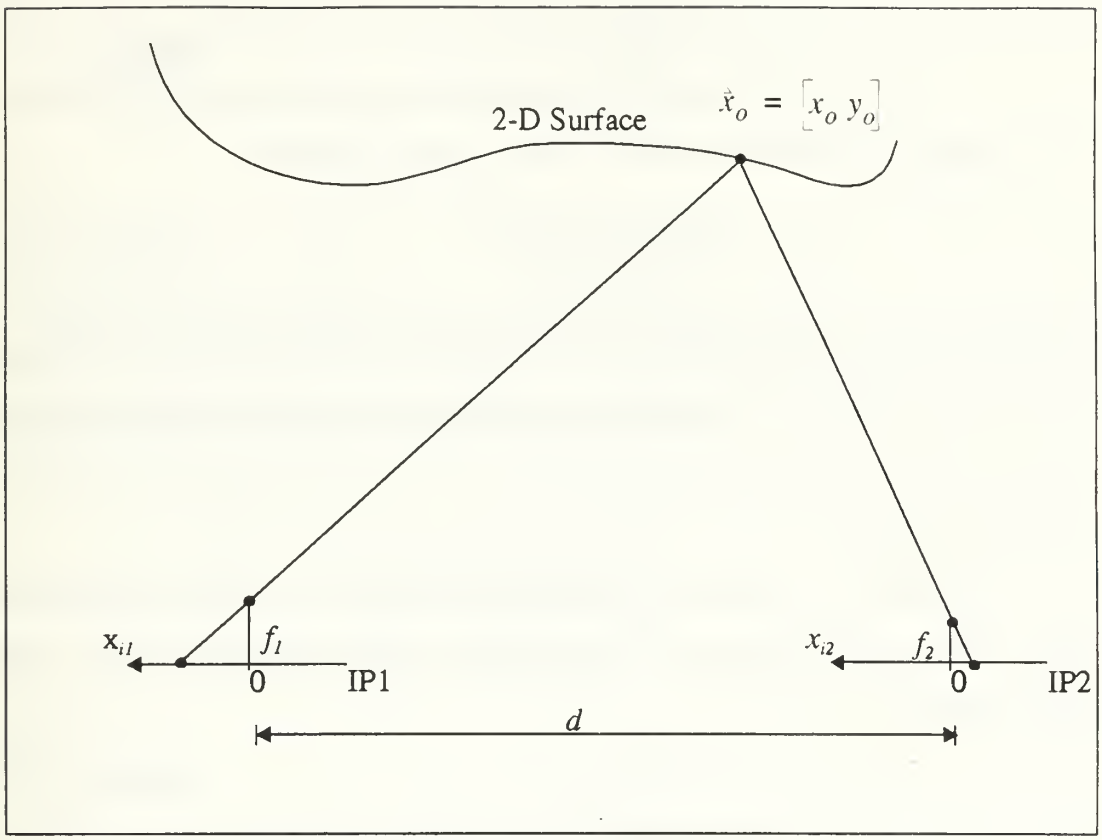

Figure 3.2 Two Dimensional Stereo Vision

Then image points in IP2, denoted x_{i2}, may be related to object points measured relatively to the global coordinate system by:

$$\begin{bmatrix} cx_{i2} \\ c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{1}{f_2} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & d \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_o \\ y_o \\ 1 \end{bmatrix}. \qquad \text{(Eq 3.19)}$$

Using equations 3.16 and 3.19 the following relationships in object coordinates are obtained:

$$x_o = \frac{x_{i1}(f_1 - y_o)}{f_1}, \qquad \text{(Eq 3.20)}$$

$$x_o = \frac{x_{i2}(f_2 - y_o)}{f_2} - d. \qquad \text{(Eq 3.21)}$$

Equating these two equation the object's depth y_o may be found:

$$y_o = \frac{f_1 f_2 (x_{i2} - x_{i1} - d)}{f_1 x_{i2} - f_2 x_i}. \qquad \text{(Eq 3.22)}$$

The object point x_o may be found, from either equation 3.20 or equation 3.21. Hence, using two image planes permits us to determine the object point depth y_o from its corresponding image points. This was proved for a 2-D surface. Next we are going to see how to do it in a 3-D environment.

Equation 3.13, defines the relationship between the scene three dimensional points and the correspondent two dimensional image points by using a matrix of transformation:

$$Q = MH_{g \to i}. \qquad \text{(Eq 3.23)}$$

$H_{g \to i}$ is the coordinate systems transformation matrix which depends on the rotations and/ or translations of the image plane coordinate system relative to the user selected global coordinate system. M is the object - image transformation matrix, which is a function of the imaging geometry and lens model. The Q matrix is a non - invertible 3×4 matrix (assuming homogeneous coordinates) and may be generically represented by:

$$Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \end{bmatrix}. \qquad \text{(Eq 3.24)}$$

The missile must have sufficient processing capability to find this matrix in real time. The object - image transformation matrix M is generally invariant (however zooming the scene, for example, changes its value). The coordinate systems transformation matrix $H_{g \to i}$ must

be dynamically updated as the missile/target engagement proceeds. The selected global coordinate system for missile guidance simulation purposes is a ground coordinate system which will be presented in the next chapter. For full dimension (3 -D) stereo vision, the two cameras arrangement may be described using homogeneous coordinates as:

$$\hat{X}_{ic} = Q^c \hat{X}_o, \qquad \text{(Eq 3.25)}$$

where the index $c = 1, 2$ refers to the cameras. Since each of these two matrix equations (one for each sensor) represents two equations in physical coordinates, we obtain four equations and three unknowns by using the two cameras stereo arrangement. The matrix equation 3.25 can be explicitly written for each sensor as:

$$\begin{bmatrix} c_1 y_{i1} \\ c_1 z_{i1} \\ c_1 \end{bmatrix} = \begin{bmatrix} q_{111} & q_{112} & q_{113} & q_{114} \\ q_{121} & q_{122} & q_{123} & q_{124} \\ q_{131} & q_{132} & q_{133} & q_{134} \end{bmatrix} \begin{bmatrix} x_o^g \\ y_o^g \\ z_o^g \\ 1 \end{bmatrix}, \qquad \text{(Eq 3.26)}$$

and,

$$\begin{bmatrix} c_2 y_{i2} \\ c_2 z_{i2} \\ c_2 \end{bmatrix} = \begin{bmatrix} q_{211} & q_{212} & q_{213} & q_{214} \\ q_{221} & q_{222} & q_{223} & q_{224} \\ q_{231} & q_{232} & q_{233} & q_{234} \end{bmatrix} \begin{bmatrix} x_o^g \\ y_o^g \\ z_o^g \\ 1 \end{bmatrix}. \qquad \text{(Eq 3.27)}$$

The index of each matrix element is composed by three numbers, the first is the camera number and the next two represent the element position into the matrix. Each of these two matrix equations generates two equations in physical coordinates. To find these equations, the arbitrary constants (c_1 and c_2) have to be calculated. Then, each of the constants is substituted into the two remaining matrix equations. Finally, regrouping terms as coefficients of x_o^g, y_o^g and z_o^g , a set of four equations is obtained. Performing this procedure to obtain the first physical equation from the matrix equation 3.26, we get:

$$c_1 = q_{131} x_o^g + q_{132} y_o^g + q_{133} z_o^g + q_{134}, \qquad \text{(Eq 3.28)}$$

substituting this expression into $c_1 y_{i1}$, and regrouping terms, we obtain:

(Eq 3.29)

$$((q_{111} - q_{131}y_{i1})x_o^g + (q_{121} - q_{132}y_{i1})y_o^g + (q_{113} - q_{133}y_{i1})z_o^g = q_{134}y_{i1} - q_{114})$$

The set of four equations and three unknowns is written compactly in matrix notation as:

$$P\vec{X}_o^g = \vec{F} \qquad \text{(Eq 3.30)}$$

or

$$\begin{bmatrix} q_{111} - q_{131}y_{i1} & q_{121} - q_{132}y_{i1} & q_{113} - q_{133}y_{i1} \\ q_{121} - q_{131}z_{i1} & q_{122} - q_{132}z_{i1} & q_{123} - q_{133}z_{i1} \\ q_{211} - q_{231}y_{i2} & q_{212} - q_{232}y_{i2} & q_{213} - q_{233}y_{i2} \\ q_{221} - q_{231}z_{i2} & q_{222} - q_{232}z_{i2} & q_{223} - q_{233}z_{i2} \end{bmatrix} \vec{X}_o^g = \begin{bmatrix} q_{134}\,y_{i1} - q_{114} \\ q_{134}\,z_{i1} - q_{124} \\ q_{234}\,y_{i2} - q_{214} \\ q_{234}\,z_{i2} - q_{224} \end{bmatrix} \qquad \text{(Eq 3.31)}$$

Equation 3.31 may be solved using least square techniques by forming the pseudoinverse of P denoted $P^{\dagger}$. Hence:

$$P\vec{X}_o^g = \vec{F} \Rightarrow P^T P\vec{X}_o^g = P^T\vec{F} \Rightarrow \vec{X}_o^g = (P^T P)^{-1}P^T\vec{F} \Rightarrow \vec{X}_o^g = P^{\dagger}\vec{F}. \quad \text{(Eq 3.32)}$$

This equation yields the mean square estimate for the object point $\vec{X}_o^g$. Alternatively, $\vec{X}_o^g$ may be found by using three of the four equations, assuming that the three equations are linearly independent.

In this exposition, we have assumed that the necessary image plane point correspondences have been determined. It is important to say that this is the most difficult problem in the development of a stereo vision algorithm. Techniques to solve this problem are presented in [Ref. 3]. [Ref.4] presents an algorithm to match stereo images.

B. ESTIMATING 3-D MOTION PARAMETERS OF A RIGID BODY FROM TWO CONSECUTIVE IMAGE FRAMES

1. General

In section A of this chapter, we have shown that using two cameras, we are able to find the object (target) 3-D position relative to a user selected global coordinate system. As intermediate steps, it is necessary to establish feature correspondences between selected points in the two stereo images (static stereo). It is also necessary to establish feature correspondences for all pairs of consecutive image frames in each camera's image sequence. The targets that we are interested in are mainly airplanes. Therefore, we may use as points for feature correspondences the tips of the wings, nose, stabilizers and rudder.

Estimation of the 3-D target acceleration components may be divided in three steps. In the first step, the target estimated points at t_1 and t_2 are used to estimate its 3-D velocity components. In the second step, the 3-D velocity components of the target are computed using the target's estimated points at t_2 and t_3. Finally, the third step estimates the target's 3-D acceleration components by identifying the time change of the target's velocity for each of the three velocity components. Alternatively, we may use Kalman filtering theory to estimate the 3-D target acceleration components.

A different approach for estimating the 3-D, time dependent, target motion is now presented. The formal definition of a rigid 3-D object is one for which the 3-D distances between any pair of points on the object do not change with time that is, for all pair of points on the 3-D object:

$$\left\| \vec{X}_m - \vec{X}_n \right\|^2 = c_{mn}, \forall t, \forall m, n; \qquad \text{(Eq 3.33)}$$

where c_{mn} are constants. The assumption of an rigid, or nondeformable target is reasonable and creates additional constraints for motion estimation. Rigidity constrains the motion of individual object points to be strongly coupled, although the need for point correspondence maintains. Then, the 3-D translation of the target may be determined by estimating the

translation parameters of a single point object. The basis of estimating the target's motion using this approach, is that the 3-D motion of a rigid body can be described by a 3-D translation vector and three rotation angles chosen with respect to a user selected coordinate system. Then, six parameters completely define the target's motion. Formulating the rotations using three rotation matrices (R_θ, R_α and R_ρ), the target's motion is described by:

$$\vec{X}_o(t_2) = R\vec{X}_o(t_1) + \vec{T}, \qquad \text{(Eq 3.34)}$$

where R is the overall rotation matrix:

$$R = R_\theta R_\alpha R_\rho, \qquad \text{(Eq 3.35)}$$

and $\vec{T}$ is the translation matrix. Equation 3.34 may be represented in homogeneous coordinates as:

$$\hat{X}_o(t_2) = \begin{bmatrix} & R & & \vdots & \vec{T} \\ - & - & - & \vdots & - \\ 0 & 0 & 0 & \vdots & 1 \end{bmatrix} \hat{X}_o(t_1). \qquad \text{(Eq 3.36)}$$

2. Monocular Motion Estimation Using Two Perspective Views

Our goal is to compute dynamically the rotation (R) and translation (T) matrices from point (or feature) correspondences between two perspective views. We can divide the process of estimating the three - dimensional motion of the target from image sequences in three steps. The first step is to establish feature correspondences between two consecutive image frames. Correspondences between features may be established through matching or inter - frame tracking. [Ref.4] develops a two - view/stereo matcher that computes displacement fields from two images. The second step is to estimate the motion parameters (R and T matrices). The third step is to estimate the 3-D motion of the target using equation 3.34. There are a number of papers in the digital image processing and computer vision literature dealing with estimation of the motion parameters of the 3-D motion of a rigid

body from two consecutive image frames. Weng, Huang and Ahaja [Ref. 5], propose an algorithm that given 8 point correspondences solves for a intermediate matrix called the essential parameter matrix (E). Then the Rotation matrix (R) and the translational direction (the unit vector $\frac{\vec{T}}{\|T\|}$) are obtained from E. The magnitude of the translational vector ($\|T\|$) and the absolute depths of the object points (x_{ok} and x'_{ok} where x_{ok} is the absolute depth of the object's k feature point and x'_{ok} is the absolute depth of the object's k feature point after being rotated and translated) cannot be determined by monocular vision. The algorithm also solves for the relative depths ($\frac{x_{ok}}{\|T\|}$ and $\frac{x'_{ok}}{\|T\|}$). The algorithm is unable to estimate the target's position because it is not possible to calculate the absolute depth. This agrees with intuition, due to the lack of invertibility of the 3-D to 2-D image transformation. To overcome this problem, we propose to estimate the absolute depth by correlating the dimensions of the target over the image plane with the guessed physical dimensions of the target. A relatively easy trigonometric approach permits us to estimate the target's depth given the target's physical dimensions. Another approach for emulating stereo vision is to use range and directional spatial information from the seeker. This information is combined with the monocular vision equations in order to estimate the target's 3-D motion.

In conclusion, monocular vision may be applied to estimate the motion of the target if additional information about the target is available (or guessed). The target's acceleration components may now be estimated and injected into the missile's control algorithm.

3. Stereo Motion Estimation Using Two Perspective Views

As we have seen, two or more spatially distributed sensors enables determination of the 3-D target motion. Here, our goal is to determine, not only the target's motion but also the rotation and translation matrices that describe the motion. Given two consecutive

time samples or "frames" in each sensor, as well as image point correspondences, we may write:

$$\hat{X}_i^1(t) = Q_{g\to i}^1 \hat{X}_o(t), \qquad \text{(Eq 3.37)}$$

$$\hat{X}_i^2(t) = Q_{g\to i}^2 \hat{X}_o(t), \qquad \text{(Eq 3.38)}$$

$$\hat{X}_i^1(t+T_s) = Q_{g\to i}^1 \hat{X}_o(t+T_s) = Q_{g\to i}^1 \left[\begin{array}{ccc|c} & R & & \vec{T} \\ - & - & - & - \\ 0 & 0 & 0 & 1 \end{array}\right] \hat{X}_o(t), \qquad \text{(Eq 3.39)}$$

$$\hat{X}_i^2(t+T_s) = Q_{g\to i}^2 \hat{X}_o(t+T_s) = Q_{g\to i}^2 \left[\begin{array}{ccc|c} & R & & \vec{T} \\ - & - & - & - \\ 0 & 0 & 0 & 1 \end{array}\right] \hat{X}_o(t). \qquad \text{(Eq 3.40)}$$

The matrix $Q_{g\to i}^1$ is the, already known, 3-D to 2-D transformation matrix (the superscript refers to the sensor). T_s is the time between two consecutive images. The system is represented in homogeneous coordinates. Assuming n point for point correspondences, a total of $8n$ equations in physical coordinates is obtained. However, the number of unknowns is $12 + 3n$ (9 elements of R, 3 elements of $\vec{T}$ and 3 elements for each $\vec{X}_o$). This yields the constraint on the number of point for point correspondences:

$$12 + 3n \le 8n. \qquad \text{(Eq 3.41)}$$

Since n must be an integer, $n \ge 3$. Thus, three corresponding image points from two views in two frames are sufficient to determine both the motion parameters and the 3-D location of the object points.

The three target acceleration components may be computed by taking the second derivative following the filtering of the target's motion data. Alternatively, the target's motion may be processed by Kalman filters to estimate its acceleration components.

This additional information about the target's behavior may be used to improve the missile guidance towards the target. In order to effectively use this information, we have first of all to determine control laws that can use and produce better results if this information is available. This will be stressed in the next chapter.

IV. SIMULATION DEVELOPEMENT

A. CONTROL ALGORITHMS DEVELOPMENT

Chapter II introduced proportional navigation guidance. In this section we derive more advance guidance laws. Contrary to PROPNAV, these guidance laws use the estimated acceleration of the target as a additional input to the homing loop. As will be seen in Chapter V, the advanced guidance laws relax the interceptor acceleration requirements and, in general, yield smaller miss distances.

1. Augmented Proportional Navigation

Proportional navigation issues control commands that are proportional to the predicted zero effort miss normal to the line of sight (ZEM_{PLOS}). That is, the missile guidance system tries to minimize the final miss distance between the target and the missile by issuing acceleration commands that are proportional to the miss distance, that would result if the missile made no further corrective acceleration and the target did not maneuver. Therefore, if the target maneuvers evasively it generates additional miss distance that is not accounted for in the PROPNAV guidance law. Augmented proportional navigation also issues guidance commands that are proportional to the predicted miss distance. However, for augmented PROPNAV, the miss distance is estimated by taking into account the maneuver of the target (target acceleration). The augmented PROPNAV target's acceleration dependent term will be calculated. This term is injected into the homing loop to enhance the guidance performance. In the following analysis, we follow the nomenclature of Chapter II and the geometry of Figure 2.6.

The x component of the miss distance, for an evasive target, is computed as follows (the y component is computed similarly):

$$\frac{d}{dt'}\,(r_x(t')) = v_x(t'), \qquad \text{(Eq 4.1)}$$

where t' represents time. Then,

$$ZEM_x(t) = r_x|_{t=t_F} = r_x(t) + \int_t^{t_F} v_x(t')\,dt'. \qquad \text{(Eq 4.2)}$$

Where, $ZEM_x(t)$ is the x component of the zero effort miss, predicted at time $t' = t$ and $r_x(t)$ is the present missile - target relative distance along the x axis. But,

$$a_t^x(t') = \frac{d}{dt'}\,(v_x(t')), \qquad \text{(Eq 4.3)}$$

where $a_t^x(t')$ is the x component of the target acceleration. Then,

$$\int_{v_x(t)}^{v_x(t')} dv_x(t') = \int_t^{t'} a_t^x(t'')\,dt'', \qquad \text{(Eq 4.4)}$$

where t'' is the variable of integration. Hence:

$$v_x(t') = v_x(t) + \int_t^{t'} a_t^x(t'')\,dt''. \qquad \text{(Eq 4.5)}$$

Substituting this equation into equation 4.2, we get the expression for the predicted x component of the ZEM at time t:

$$ZEM_x(t) = r_x|_{t=t_F} = r_x(t) + v_x(t)\,t_{go} + \int_t^{t_F}\int_t^{t'} a_t^x(t'')\,dt''\,dt''. \qquad \text{(Eq 4.6)}$$

Where, $t_{go} = t_F - t$ is the time to go. The y component of the $ZEM(t)$ is obtained using the same reasoning, and is:

$$ZEM_y(t) = r_y|_{t=t_F} = r_y(t) + v_y(t)\,t_{go} + \int_t^{t_F}\int_t^{t'} a_t^y(t'')\,dt''\,dt''. \qquad \text{(Eq 4.7)}$$

The two interior integrals $\int_t^{t'} a_t^x(t'')\,dt''$ and $\int_t^{t'} a_t^y(t'')\,dt''$ are time functions; call:

$$k_x(t') = \int_t^{t'} a_t^x(t'')\,dt'', \qquad \text{(Eq 4.8)}$$

and

$$k_y(t') = \int_t^{t'} a_t^y(t'')\,dt''. \qquad \text{(Eq 4.9)}$$

Then, equations 4.6 and 4.7 may be expressed as:

, (Eq 4.10)

$$ZEM_x(t) = r_{x_{t=t_F}} = r_x(t) + v_x(t)\,t_{go} + \int_t^{t_F} k_x(t'')\,dt'' = r_x(t) + v_x(t)\,t_{go} + h_x(t_{go})$$

(Eq 4.11)

$$ZEM_y(t) = r_{y_{t=t_F}} = r_y(t) + v_y(t)\,t_{go} + \int_t^{t_F} k_y(t'')\,dt'' = r_y(t) + v_y(t)\,t_{go} + h_y(t_{go});$$

where $h_x(t_{go})$ and $h_y(t_{go})$ are t_{go} and maneuver dependent functions. The component of the ZEM that is perpendicular to the LOS, ZEM_{PLOS}, is:

$$ZEM_{PLOS}(t) = ZEM_y(t)\cos(\lambda(t)) - ZEM_x(t)\sin(\lambda(t)). \qquad \text{(Eq 4.12)}$$

Substituting 4.10 and 4.11 into this equation and setting $\cos(\lambda(t)) = \frac{r_x(t)}{r(t)}$ and $\sin(\lambda(t)) = \frac{r_y(t)}{r(t)}$, we obtain:

$\mathrm{ZEM}_{PLOS}(t) =$ (Eq 4.13)

$$(r_y(t) + v_y(t)\,t_{go} + h_y(t_{go}))\frac{r_x(t)}{r(t)} - (r_x(t) + v_x(t)\,t_{go} + h_x(t_{go}))\frac{r_y(t)}{r(t)}.$$

From equation 2.52:

$$\dot{\lambda}(t) = \frac{r_x(t)\,v_y(t) - r_y(t)\,v_x(t)}{r^2(t)}. \qquad \text{(Eq 4.14)}$$

Then equation 4.13, which takes into account the target's acceleration to estimate the miss distance, may be reduced to:

$$ZEM_{PLOS}(t) = t_{go} r(t) \dot{\lambda}(t) + (h_y(t_{go}) \frac{r_x(t)}{r(t)} - h_x(t_{go}) \frac{r_y(t)}{r(t)}) . \quad \text{(Eq 4.15)}$$

The PROPNAV guidance command may be expressed in terms of the ZEM as (see derivation in Chapter II):

$$u_m(t) = \frac{NV_c(t)\,\mathrm{ZEM}_{\mathrm{PLOS}}(t)}{r(t)\,t_{go}}, \quad \text{(Eq 4.16)}$$

where $ZEM_{PLOS}(t)$ was then derived for a nonresponsive target. If the target maneuvers the zero effort miss is augmented by an additional term, on the right hand side of equation 4.15. Therefore, a perfectly plausible guidance law, in the presence of target maneuver, would be:

(Eq 4.17)

$$u_m|_{APN}(t) = NV_c(t)\dot{\lambda}(t) + \frac{NV_c(t)}{r(t)\,t_{go}} (h_y(t_{go}) \cos(\lambda(t)) - h_x(t_{go}) \sin(\lambda(t))) .$$

This guidance law is PROPNAV with an extra term that accounts for the maneuver of the target. The equation was derived for a nonlinearized geometry. The impossibility of knowing, a priori, the future target maneuver, precludes the calculation of $h_y(t_{go})$, $h_x(t_{go})$ and t_{go}. However, if the target desires to inflict the most miss distance it must maneuver at a small time to go. Also, considering the time constants associated with the target's maneuver, we propose to approximate the time dependent target acceleration $a_t(t')$ by a constant target acceleration($a_t(t)$) i.e.assume $a_t(t') = a_t(t)$. Then, the control law may be stated as:

(Eq 4.18)

$$u_m|_{APN}(t) = NV_c(t)\dot{\lambda}(t) + \frac{NV_c(t)}{r(t)\,t_{go}} \left(\frac{a_t^y(t)\,t_{go}^2}{2} \cos(\lambda(t)) - \frac{a_t^x(t)\,t_{go}^2}{2} \sin(\lambda(t)) \right)$$

or,

$$u_m|_{APN} = NV_c\dot{\lambda} + \frac{NV_c t_{go}}{2r} (a_t^y \cos(\lambda) - a_t^x \sin(\lambda)) ; \quad \text{(Eq 4.19)}$$

where the time factor was dropped. The extra term, present in augmented PROPNAV (expression in parenthesis on the right hand side of equation 4.19), is proportional to the component of the target acceleration normal to the LOS.

Linearization of the nonlinear missile - target geometry is shown in [Ref. 1] to be an accurate approximation to the actual geometry. Then, assuming a linearized geometry, equation 4.19 may be further reduced to:

$$u_m = NV_c\dot{\lambda} + \frac{N}{2}a_t^y; \qquad \text{(Eq 4.20)}$$

where we have considered that the LOS angle is small. A zero - lag augmented proportional navigation homing loop assuming linearized geometry is shown in Figure 4.1.

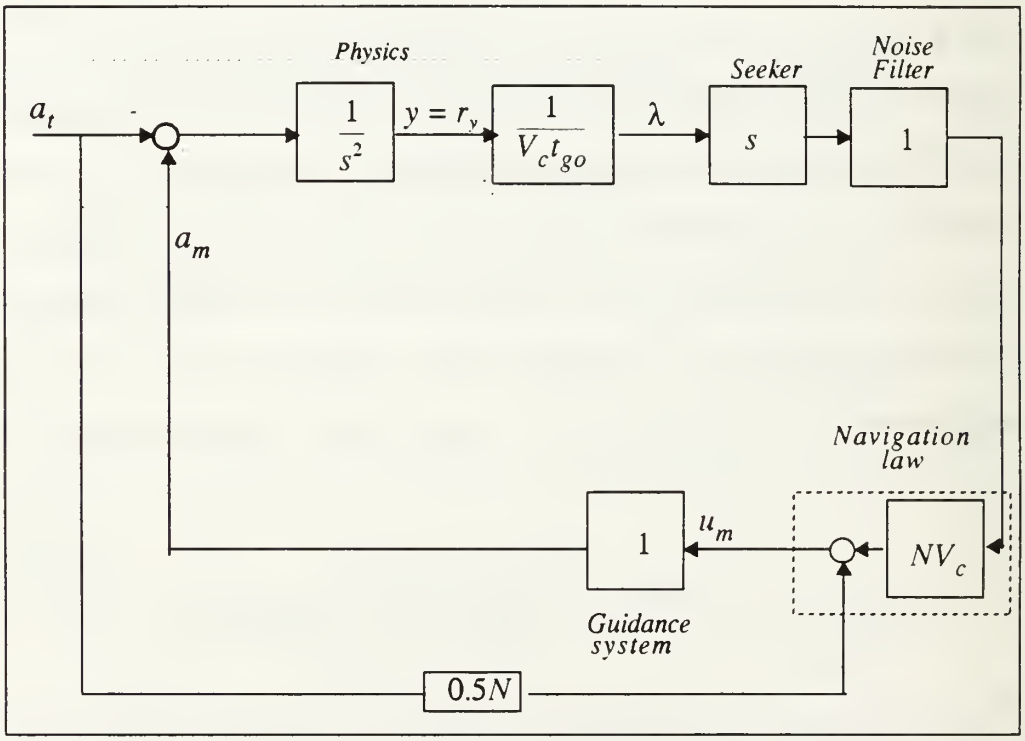

Figure 4.1 Zero - lag Augmented Proportional Navigation Homing Loop

The additional target maneuver term required by the guidance law, appears as a feedforward term in the homing loop block diagram.

2. Optimal Intercept Guidance

The missile - target engagement scenario may be described in state space representation by the following linear system:

$$\dot{x}(t) = f[x(t), u(t), t] = Ax(t) + Bu(t); \qquad \text{(Eq 4.21)}$$

x is the n - dimensional state vector describing the relative movement between the missile and the target and also, the dynamics of the guidance system. The variable u is the m - dimensional missile's control input vector. We seek to find a guidance law that is a function of the system states. There is an infinite number of possible guidance laws. Thus, it is necessary to state in mathematical terms what the guidance law should do. Certainly we wish to design a terminal controller that would bring certain components of $x(t_F)$ to zero, using "acceptable" levels of control. One way to do this is to minimize a performance index made up of a quadratic form in the control:

$$J = \int_0^{t_F} L[x(t), u(t), t]\, dt = \frac{1}{2}\int_0^{t_F} u^2(t)\, dt; \qquad \text{(Eq 4.22)}$$

subject to the terminal constraint:

$$x_i(t_F) = 0, \quad i = 0, 1, \ldots, p, \qquad \text{(Eq 4.23)}$$

and the constraints:

$$\dot{x}(t) = f[x(t), u(t), t] = Ax(t) + Bu(t), \qquad \text{(Eq 4.24)}$$

$$x(0), given. \qquad \text{(Eq 4.25)}$$

In equation 4.23, $p \leq n$.

The miss distance will always be zero in a zero - lag PROPNAV navigation homing loop. Guidance system lags or subsystem dynamics will cause miss distance. Optimal guidance eliminates miss distance by canceling out the guidance system dynamics. In this way the optimal guidance law attempts to make the real world guidance system

appear to be a "perfect" (zero - lag) guidance system. To find the optimal control vector, $u(t)$, that brings the system from a initial state $x(0)$ to a terminal state $x(t_F)$ (where some of its components are zero), we can use the method of the Lagrange multipliers. Then the constraints (4.23) and (4.24) may be adjoined to the performance function (4.22) by using the multipliers $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_p, 0_{p+1} \ldots, 0_n)^T$ and $\lambda = (\lambda_1, \ldots, \lambda_n)^T$ as follows:

$$\bar{J} = \varepsilon^T x(t_F) + \frac{1}{2}\int_0^{t_F} \{u^2(t) + \lambda^T(t)\,[Ax(t) + Bu(t) - \dot{x}(t)]\}\,dt. \quad \text{(Eq 4.26)}$$

The Hamiltonian is defined as follows:

$$H[x(t), u(t), t] = L[x(t), u(t), t] + \lambda^T f[x(t), u(t), t]. \quad \text{(Eq 4.27)}$$

Integrating the last term on the right hand side of equation 4.26 yields:

(Eq 4.28)

$$\bar{J} = \varepsilon^T x(t_F) - \lambda^T(t_F)\,x(t_F) + \lambda^T(0)\,x(0) + \int_0^{t_F} \{H[x(t), u(t), t] + \dot{\lambda}(t)\,x(t)\}\,dt.$$

Considering the variation in $\bar{J}$ due to variations in the control vector $u(t)$, we get:

(Eq 4.29)

$$d\bar{J} = [(\varepsilon^T - \lambda^T)\,dx]_{t=t_F} + [\lambda^T dx]_{t=0} + \int_0^{t_F} \left[\left(\frac{\partial H}{\partial x} + \dot{\lambda}^T(t)\right) dx + \frac{\partial H}{\partial u} du \right] dt$$

In order to make the variations in $\bar{J}$ due to variations in $u(t)$ independent from the variations in $x(t)$ produced by the variations in $u(t)$, we choose the influence functions $\lambda(t)$ to cause the coefficients of dx to vanish:

$$\dot{\lambda}^T(t) = -\frac{\partial H}{\partial x} = -\frac{\partial L}{\partial x} - \lambda^T \frac{\partial f}{\partial x}. \quad \text{(Eq 4.30)}$$

Then:

$$\dot{\lambda}(t) = -A^T \lambda(t), \quad \text{(Eq 4.31)}$$

with boundary conditions:

$$\lambda(t_F) = \varepsilon. \quad \text{(Eq 4.32)}$$

Using these results, equation 4.29 becomes:

$$d\bar{J} = \lambda^T(0)\,dx(0) + \int_0^{t_F} \frac{\partial H}{\partial u}\,du\,dt. \qquad \text{(Eq 4.33)}$$

Hence, $\lambda^T(0)$ is the gradient of J with respect to variations in the initial conditions, while holding $u(t)$ constant and satisfying the constraints of the problem. For an extremum, $d\bar{J}$ must be zero for arbitrary $du(t)$. This can only happen if:

$$\frac{\partial H}{\partial u} = 0, \quad 0 \le t \le t_F, \qquad \text{(Eq 4.34)}$$

or:

$$u^T + \lambda^T B = 0. \qquad \text{(Eq 4.35)}$$

Then, we may determine the control vector $u(t)$, as:

$$u(t) = -B^T\lambda(t). \qquad \text{(Eq 4.36)}$$

Substituting equation 4.36 into equation 4.24 and repeating equation 4.31, the following two - point boundary value problem is obtained:

$$\begin{bmatrix} \dot{x} \\ \dot{\lambda} \end{bmatrix} = \begin{bmatrix} A & -BB^T \\ 0 & -A^T \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix}. \qquad \text{(Eq 4.37)}$$

The $2n$ boundary conditions are:

$$x(0), \quad given, \qquad \text{(Eq 4.38)}$$

$$x_i(t_F) = 0, \qquad i = 1, \ldots, p; \qquad \text{(Eq 4.39)}$$

$$\lambda_i(t_F) = 0, \qquad i = p+1, \ldots, n. \qquad \text{(Eq 4.40)}$$

The n boundary conditions 4.39 and 4.40 may be replaced by the boundary condition 4.32, which may be rewritten as:

$$\lambda_i(t_F) = \varepsilon_i; \qquad i = 0, \ldots, p, \qquad \text{(Eq 4.41)}$$

$$\lambda_i(t_F) = 0; \qquad i = p+1, \ldots, n. \qquad \text{(Eq 4.42)}$$

The two - point boundary value problem 4.37, 4.38, 4.41 and 4.42 may be solved by the sweep method [Ref. 6]. The sweep method seeks to find solutions of the form:

$$\lambda(t) = W(t)x(t) + Y(t)\varepsilon; \qquad \text{(Eq 4.43)}$$

$$z = U(t)x(t) + V(t)\varepsilon, \qquad \text{(Eq 4.44)}$$

where $W(t), Y(t), U(t)$ and $V(t)$ are time dependent matrices, $z^T = [x_1, \ldots, x_p]\big|_{t=t_F}$ and $\varepsilon^T = [\varepsilon_1, \ldots, \varepsilon_p] = [\lambda_1, \ldots, \lambda_p]\big|_{t=t_F}$. Therefore, we want to find solutions for the influence functions $\lambda(t)$ that are function of the state vector $x(t)$ and the final value of the influence functions, or equivalently, of the specified final states z. Since equations 4.43 and 4.44 must be valid at $t = t_F$:

$$W(t_F) = 0, \qquad \text{(Eq 4.45)}$$

$$V(t_F) = 0, \qquad \text{(Eq 4.46)}$$

$$Y(t_F) = \left[\begin{array}{c} I_{p\times p} \\ - \\ 0_{(n-p)\times p} \end{array}\right]_{n\times p}, \qquad \text{(Eq 4.47)}$$

$$U(t_F) = \left[\begin{array}{c|c} I_{p\times p} & 0_{p\times(n-p)} \end{array}\right]_{p\times n}; \qquad \text{(Eq 4.48)}$$

where I is the identity matrix and 0 is a zero matrix with the specified dimensions. Substituting equation 4.43 into 4.37 and treating ε as a constant vector, we get:

$$\dot{W}x + W\dot{x} + \dot{Y}\varepsilon = -A^T(Wx + Y\varepsilon). \qquad \text{(Eq 4.49)}$$

Substituting $\dot{x}$ from equation 4.37 into the last equation, and again using equation 4.43 to eliminate λ, we obtain:

$$(\dot{W} + WA + A^TW - WBB^TW)x + (A^TY + \dot{Y} - WBB^TY)\varepsilon = 0. \qquad \text{(Eq 4.50)}$$

This expression must be true for any x and ε, so the coefficients of x and ε must vanish:

$$\dot{W} + WA + A^T W - WBB^T W = 0; \qquad W(t_F) = 0, \qquad \text{(Eq 4.51)}$$

and

$$\dot{Y} + A^T Y - WBB^T Y = 0; \qquad Y(t_F) = \begin{bmatrix} I_{p\times p} \\ - \\ 0_{n-p\times p} \end{bmatrix}_{n\times p} \qquad \text{(Eq 4.52)}$$

Next, we differentiate equation 4.44 with respect to time, treating ε and z as constant vectors:

$$\dot{U}x + U\dot{x} + \dot{V}\varepsilon = 0. \qquad \text{(Eq 4.53)}$$

Substituting $\dot{x}$ from equation 4.37, and using equation 4.43 to eliminate λ, we obtain:

$$(\dot{U} + UA - UBB^T W)\,x + (\dot{V} - UBB^T Y)\,\varepsilon = 0. \qquad \text{(Eq 4.54)}$$

This last expression must be true for any x and ε, so the coefficients of x and ε must vanish:

$$\dot{U} + UA - UBB^T W = 0, \qquad \text{(Eq 4.55)}$$

$$\dot{V} - UBB^T Y = 0. \qquad \text{(Eq 4.56)}$$

From equations 4.52 and 4.55 and the boundary conditions 4.47 and 4.48, we conclude that:

$$U(t) = Y^T(t). \qquad \text{(Eq 4.57)}$$

Then equation 4.56 may be rewritten as:

$$\dot{V} = Y^T BB^T Y; \qquad V(t_F) = 0. \qquad \text{(Eq 4.58)}$$

The Ricatti equations 4.51, 4.52 and 4.58 may be integrated backwards from the final conditions to yield $W(t)$, $Y(t)$ and $V(t)$. The equation 4.44 is solved for ε to yield:

$$\varepsilon = [V(t)]^{-1}[z - Y^T(t)\,x(t)]. \qquad \text{(Eq 4.59)}$$

Our goal is, to find the influence functions $\lambda(t)$ in order to find the optimal control vector using equation 4.36. Equation 4.59 may be substituted into equation 4.43 to find $\lambda(t)$:

$$\lambda(t) = (W - YV^{-1}Y^T)\,x(t) + YV^{-1}z. \qquad \text{(Eq 4.60)}$$

However, the Ricatti equation 4.51 has as solution:

$$W(t) = 0. \qquad \text{(Eq 4.61)}$$

Hence, equations 4.52 and 4.58 become simply:

$$\dot{Y} + A^T Y = 0; \qquad Y(t_F) = \left[\begin{array}{c} I_{p\times p} \\ - \\ 0_{n-p\times p} \end{array}\right]_{n\times p}, \qquad \text{(Eq 4.62)}$$

and

$$V(t) = -\int_{t}^{t_F} (Y^T B B^T Y)\,dt. \qquad \text{(Eq 4.63)}$$

Combining equations 4.36, 4.60 and 4.61 yields:

$$u(t) = -B^T\lambda(t) = (-B^T Y V^{-1})\,[z - Y^T x(t)]. \qquad \text{(Eq 4.64)}$$

The final condition that we are interested on is $z = [x_1, \ldots, x_p]^T_{t=t_F} = [0, \ldots, 0]^T$. Hence the expression 4.64 may be reduced to:

$$u(t) = B^T Y V^{-1} Y^T x(t), \qquad \text{(Eq 4.65)}$$

where Y and V are computed from equations 4.62 and 4.63, respectively.

Now that we have derived the terminal controller optimal feedback guidance law, we proceed to derive the continuous feedback law described by equation 4.65 for a single lag guidance system. The single - lag guidance system is mathematically described in the Laplace domain as:

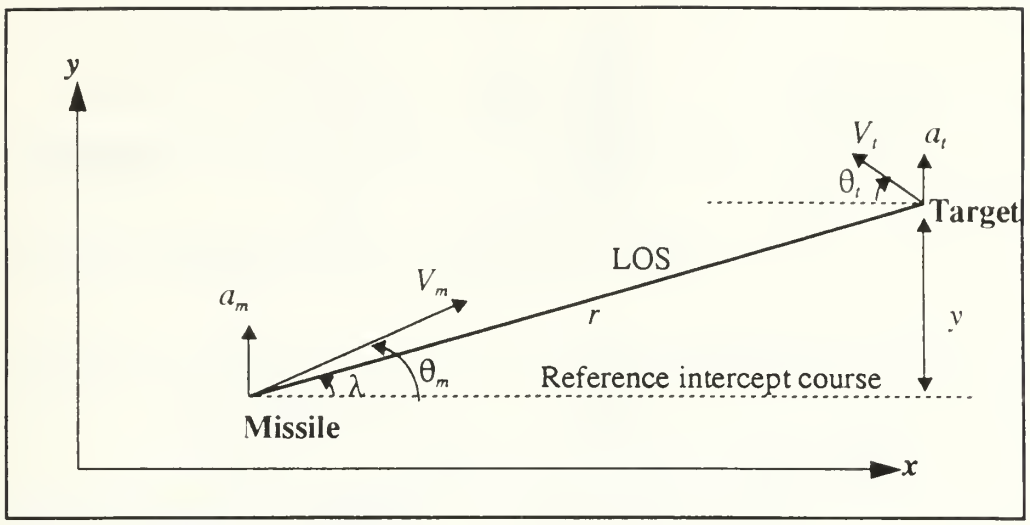

Figure 4.2 Intercept Geometry

$$\frac{a_m}{u_m} = \frac{1}{1+Ts}, \qquad \text{(Eq 4.66)}$$

where a_m is the missile's acceleration, u_m is the command acceleration, and T is the effective guidance system time constant.

The relative motion between the target and missile is considered with the linearized (small angles approximation) intercept geometry shown in Figure 4.2. The assumption of small angles (flight path angles θ_m, θ_t and LOS angle λ) permits us to express the equations of motion in terms of state variables normal to the reference intercept course. The single - lag guidance model shown in Figure 4.3 integrates the missile - target relative motion of Figure 4.2 with the dynamics of the guidance system. The diagram of blocks of Figure 4.3 can be expressed in state space form as:

$$\begin{bmatrix} \dot{y} \\ \ddot{y} \\ \dot{a}_t \\ \dot{a}_m \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{T} \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \\ a_t \\ a_m \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{T} \end{bmatrix} u_m. \qquad \text{(Eq 4.67)}$$

Thus:

$$x = \begin{bmatrix} y \\ \dot{y} \\ a_t \\ a_m \end{bmatrix}, \qquad A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{T} \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{T} \end{bmatrix}. \qquad \text{(Eq 4.68)}$$

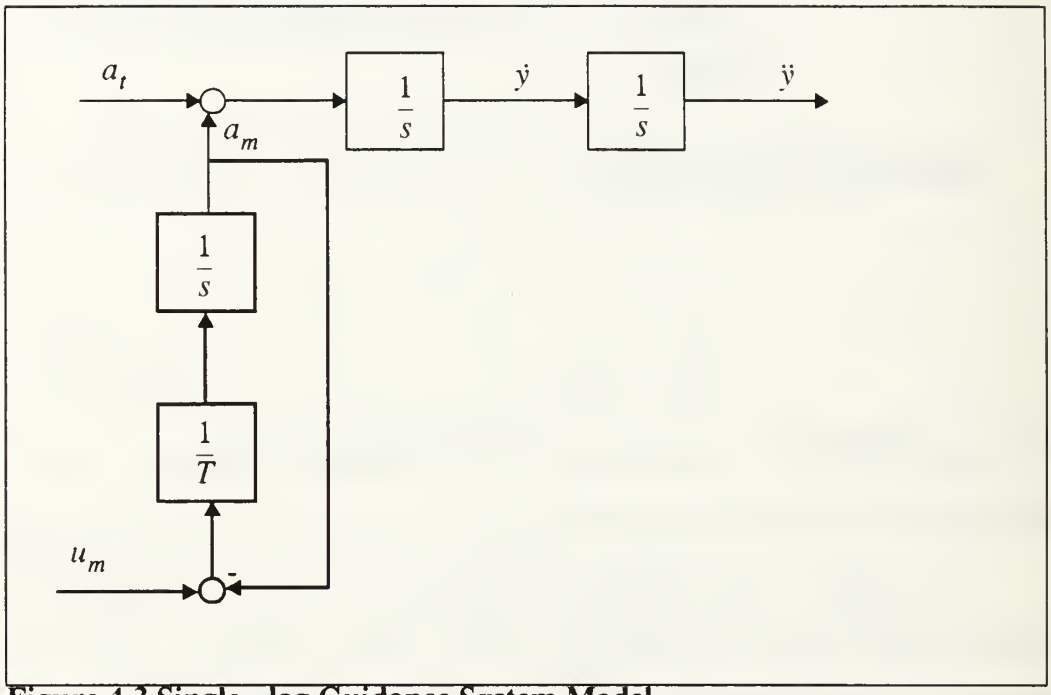

Figure 4.3 Single - lag Guidance System Model

To find the optimal feedback law from equation 4.65, Y and V have to be calculated using equations 4.62 and 4.63, respectively. The solution for Y is obtained from:

$$\dot{Y} + A^T Y = 0, \qquad Y(t_F) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}; \ , \qquad \text{(Eq 4.69)}$$

$$\dot{Y} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & -\frac{1}{T} \end{bmatrix} Y = 0, \quad Y(t_F) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \qquad \text{(Eq 4.70)}$$

Y is obtained by integrating equation 4.70 backwards, and is:

$$Y = \begin{bmatrix} 1 \\ t_F - t \\ \frac{(t_F - t)^2}{2} \\ T^2\left[1 - \frac{(t_F - t)}{T} - e^{\frac{(t_F - t)}{T}}\right] \end{bmatrix} = \begin{bmatrix} 1 \\ t_{go} \\ \frac{t_{go}^2}{2} \\ -T^2\left[e^{-\frac{t_{go}}{T}} + \frac{t_{go}}{T} - 1\right] \end{bmatrix}. \qquad \text{(Eq 4.71)}$$

The matrix V (in this case, V is a scalar since we are specifying only one terminal constraint, namely zero miss distance: $y = 0$), is computed using equation 4.63:

(Eq 4.72)

$$V(t) = -\int_t^{t_F} Y^T B B^T Y dt = (T^2) \int_t^{t_F} \left(e^{-\frac{t_{go}}{T}} + \frac{t_{go}}{T} - 1\right)^2 dt; \quad \text{where} \quad t_{go} = t_F - t.$$

After some cumbersome computations, we find:

(Eq 4.73)

$$V(t) = \frac{T^3}{3}(-k^3 + 3k^2 - 3k) + 12ke^{-k} + 3e^{-2k} - 3; \quad \text{where} \quad k = \frac{t_{go}}{T}.$$

Then using equation 4.65 we obtain the optimal feedback control law:

$$u(t) = \frac{N}{t_{go}^2}\left[y + \dot{y}t_{go} + 0.5a_t t_{go}^2 + (-a_m T^2)(e^{-k} + k - 1)\right], \qquad \text{(Eq 4.74)}$$

$$\text{where } N = \frac{6k^2(e^{-k} + k - 1)}{2k^3 - 6k^2 + 6k + 3 - 12ke^{-k} - 3e^{-2k}}; \quad k = \frac{t_{go}}{T}. \qquad \text{(Eq 4.75)}$$

It is desirable to express the state variables y and $\dot{y}$ in terms of the line of sight rate $\dot{\lambda}$. Assuming small angles, we find from Figure 4.2 that:

$$(\lambda = \frac{y}{r} = \frac{y}{V_c(t_F - t)}) \Rightarrow \dot{\lambda} = \frac{y + \dot{y}t_{go}}{V_c t_{go}^2}, \qquad \text{(Eq 4.76)}$$

hence, the optimal control law can be expressed as:

$$u(t) = NV_c\dot{\lambda} + \frac{N}{k^2}(e^{-k} + k - 1)a_m + \frac{N}{2}a_t. \qquad \text{(Eq 4.77)}$$

This equation is a biased proportional navigation guidance law where a time varying navigation gain N and an acceleration feedback path provide compensation for the missile time lag. The acceleration command is issued normal to the LOS.

B. TRIDIMENSIONAL MISSILE/TARGET ENGAGEMENT

In this section we are going to model the missile/target tridimensional engagement scenario. Three guidance laws, namely: PROPNAV, augmented PROPNAV and optimal guidance will be used and tested in missile guidance. The engagement for the two later guidance laws will be modeled, by assuming the presence of a seeker and a camera aboard the missile to extract the target's 3-D acceleration. The three dimensional MATLAB programs are presented in Appendices A through C. The simulation results are presented in the next chapter.

1. 3-D Missile /Target Geometry

The tridimensional scenario is developed in spherical coordinates by defining two perpendicular planes in pitch and yaw, as illustrated by Figure 4.4.

In Figure 4.4 r is the relative distance between missile and target; λ_{pitch} and λ_{yaw} are the line of sight angles over the pitch and yaw planes, respectively, and may be expressed as:

$$\lambda_{pitch} = \text{atan}\left[\frac{(z_t - z_m)}{\sqrt{(x_t - x_m)^2 + (y_t - y_m)^2}}\right], \qquad \text{(Eq 4.78)}$$

$$\lambda_{yaw} = \text{atan}\left[\frac{(y_t - y_m)}{(x_t - x_m)}\right]. \qquad \text{(Eq 4.79)}$$

The coordinate system shown in Figure 4.4 translates with the missile. To track the missile target tridimensional positions we define a ground based coordinate system shown in Figure 4.5.

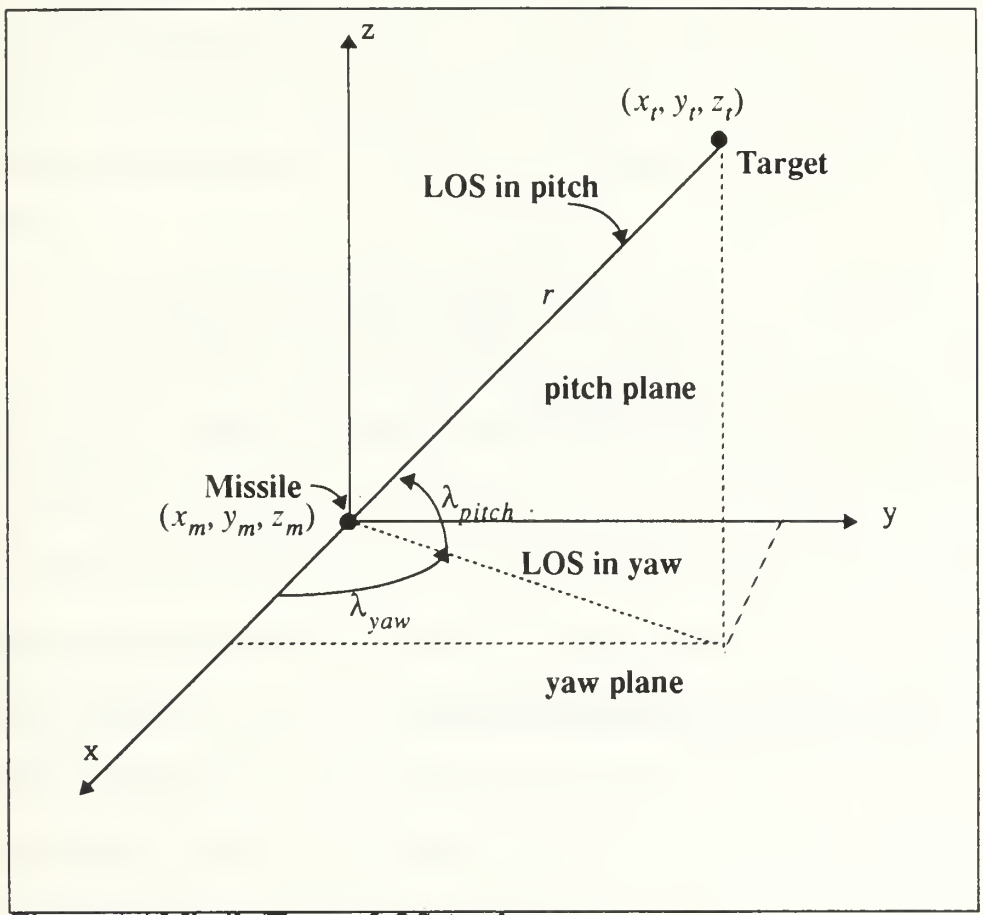

Figure 4.4 Missile/Target LOS Angles

In Figure 4.5:

$$\lambda_{m\_pitch} = \text{atan}\left(\frac{z_m}{\sqrt{x_m^2 + y_m^2}}\right), \qquad \text{(Eq 4.80)}$$

$$\lambda_{t\_pitch} = \text{atan}\left(\frac{z_t}{\sqrt{x_t^2 + y_t^2}}\right), \qquad \text{(Eq 4.81)}$$

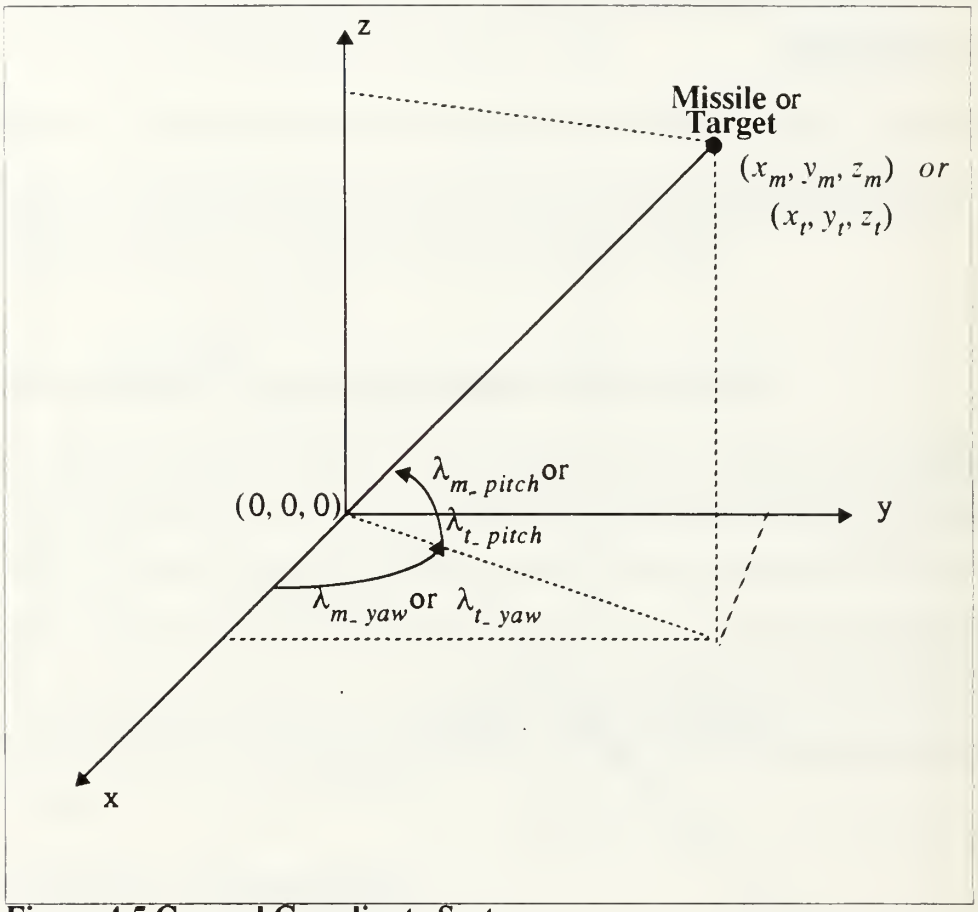

Figure 4.5 Ground Coordinate System

$$\lambda_{m\_yaw} = \text{atan}\left(\frac{y_m}{x_m}\right), \qquad \text{(Eq 4.82)}$$

$$\lambda_{t\_yaw} = \text{atan}\left(\frac{y_t}{x_t}\right). \qquad \text{(Eq 4.83)}$$

The missile is controlled in 3-D space by issuing guidance commands in two orthogonal planes (pitch and yaw), u_{pitch} and u_{yaw}. The magnitude of these commands

depends on the selected guidance law and their tridimensional direction is perpendicular to the LOS defined in the two planes (see Figure 4.4). The LOS in pitch is defined by the imaginary 3-D line from the missile to the target in the pitch plane. The LOS in yaw is defined by the imaginary 2-D line from the missile to the projection of the target over the yaw plane. The yaw plane is simply the horizontal xy plane. The pitch plane is the vertical plane normal to the horizontal plane and rotated by the yaw angle λ_{yaw}.

The guidance laws under study can be expressed as a function of the classical proportional navigation guidance law plus a term that may depend, among other variables, on the target and missile accelerations. Hence, each guidance law (classical proportional navigation, augmented proportional navigation and optimal guidance) can, in general, be expressed, as:

$$u_m(t) = NV_c\dot{\lambda} + f(a_m, a_t, t_{go}, T), \qquad \text{(Eq 4.84)}$$

where $u_m(t)$ is the guidance command that is issued perpendicular to either the LOS in yaw or the LOS in pitch. N is constant for PROPNAV and augmented PROPNAV. For optimal guidance, N is function of the time to go and the effective time constant of the guidance system. The closing speed V_c, is the relative speed between the target and the missile along either the LOS in yaw or the LOS in pitch. The LOS rate $\dot{\lambda}$ may be the LOS rate in either the pitch or the yaw planes. The term $f(a_m, a_t, t_{go}, T)$ may be function of both the missile's acceleration a_m and the target's acceleration a_t, the time to go t_{go} and the guidance system's effective time constant T. In order to generate the pitch and yaw guidance commands $u_{pitch}(t)$ and $u_{yaw}(t)$, it is first necessary to explain how to obtain the variables that they depend on.

2. Seeker Head Modeling

The seeker is able to detect, acquire and track by sensing and processing the radiation or reflection of energy by the target. The seeker is normally located in the

missile's nose and mounted on a gimballed platform which maintains the target within the field of view by rotating the platform.

The control torque to the seeker may be described by the following equation.

$$T = I\ddot{\beta}, \qquad \text{(Eq 4.85)}$$

where T is the applied torque, I is the seeker's moment of inertia and $\ddot{\beta}$ is the seeker's angular acceleration. The seeker's dynamics is modeled by the following second order differential equation:

$$\ddot{\beta} = \frac{T}{I} = -c_1(\beta - \lambda) - c_2\dot{\beta}, \qquad \text{(Eq 4.86)}$$

where the coefficients c_1 and c_2 are determined by the seeker's time constant (τ_{sk}) and damping ratio. Taking the Laplace transform of equation 4.86, assuming zero initial conditions, we obtain the filter's transfer function that represents the relationship between the LOS angle input $\lambda(s)$ and the seeker head angle output $\beta(s)$:

$$\frac{\beta(s)}{\lambda(s)} = \frac{c_1}{s^2 + c_2 s + c_1}. \qquad \text{(Eq 4.87)}$$

Assuming a damping ratio of one, the transfer function in equation 4.87 may be rewritten as:

$$\frac{\beta(s)}{\lambda(s)} = \frac{c_1}{s^2 + c_2 s + c_1} = \frac{c_1}{(s + \frac{1}{\tau_{sk}})^2}. \qquad \text{(Eq 4.88)}$$

Choosing τ_{sk} = 0.1 sec (which is a good approximation of a real world system), the constants c_1 and c_2 may be obtained:

$$c_1 = (\frac{1}{\tau_{sk}})^2 = 100, \qquad \text{(Eq 4.89)}$$

$$c_2 = 2(\frac{1}{\tau_{sk}}) = 20. \qquad \text{(Eq 4.90)}$$

Given that we are interested in the 3-D missile/target engagement, the seeker must provide line of sight rate information in both planes. Hence:

$$\frac{\beta_{pitch}}{\lambda_{pitch}} = \frac{20}{s^2 + 20s + 100}, \qquad \text{(Eq 4.91)}$$

$$\frac{\beta_{yaw}}{\lambda_{yaw}} = \frac{20}{s^2 + 20s + 100} \qquad \text{(Eq 4.92)}$$

where β_{pitch} and β_{yaw} are the seeker's pitch and yaw angles, respectively. Figures 4.6 and 4.7 depict the pitch and yaw signal flow graphs of the seeker.

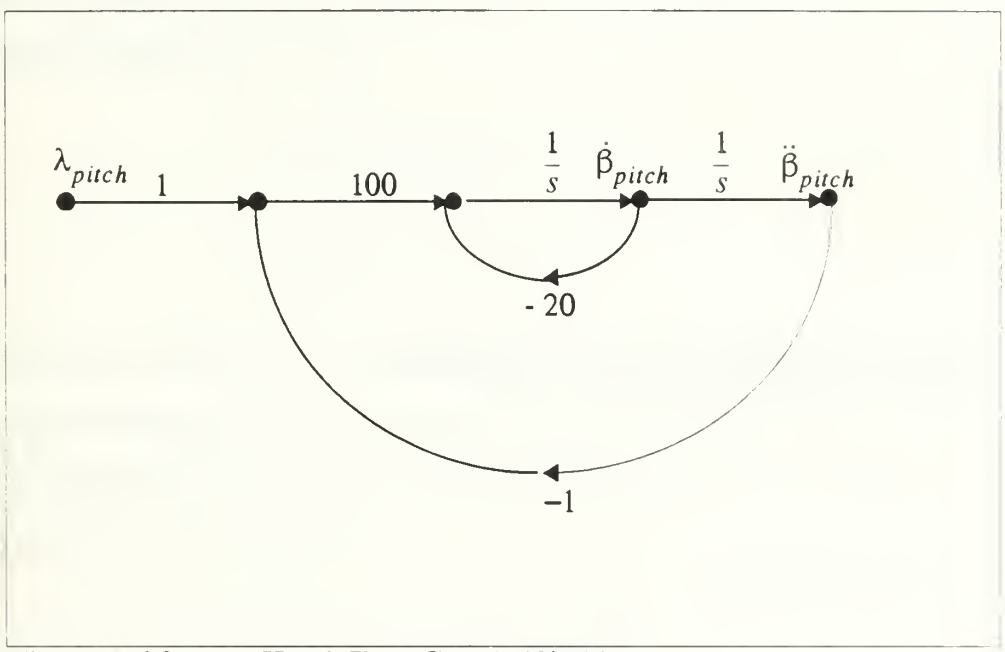

Figure 4.6 Seeker Head Flow Graph (Pitch)

From these diagrams, the continuous-time state equations of the form:

$$\dot{x}_{sk} = A_{sk}x_{sk} + B_{sk}u_{sk}, \qquad \text{(Eq 4.93)}$$

may be easily obtained. Selecting the state vector to be:

$$x_{sk} = \begin{bmatrix} \beta_{pitch} \\ \dot{\beta}_{pitch} \\ \beta_{yaw} \\ \dot{\beta}_{yaw} \end{bmatrix}, \qquad \text{(Eq 4.94)}$$

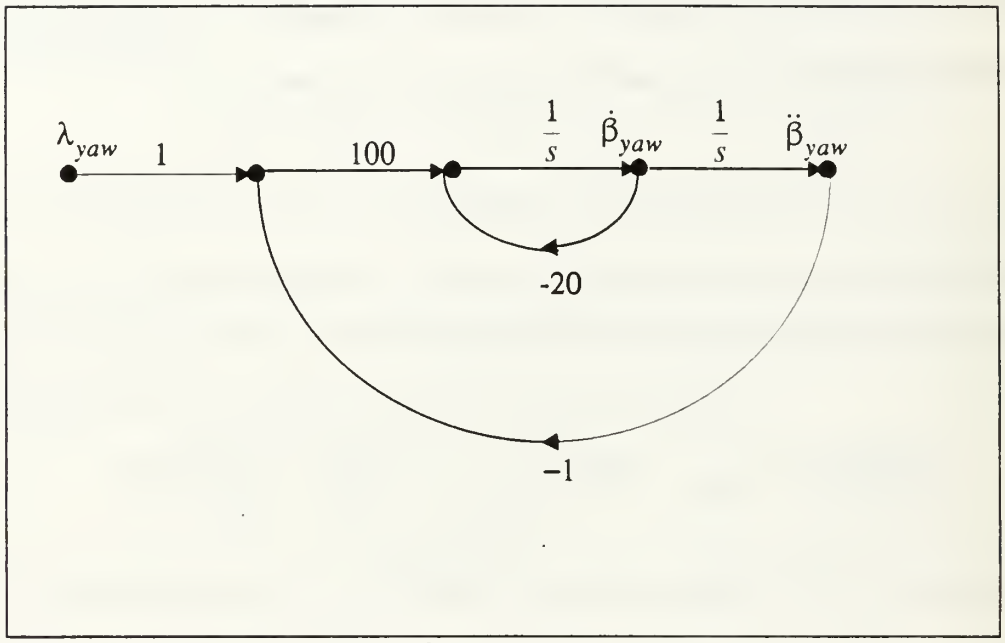

Figure 4.7 Seeker Head Flow Graph (Yaw)

and the seeker head input as:

$$u_{sk} = \begin{bmatrix} \lambda_{pitch} \\ \lambda_{yaw} \end{bmatrix}, \qquad \text{(Eq 4.95)}$$

equation 4.93 becomes:

$$\dot{x}_{sk} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -100 & -20 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -100 & -20 \end{bmatrix} x_{sk} + \begin{bmatrix} 0 & 0 \\ 100 & 0 \\ 0 & 0 \\ 0 & 100 \end{bmatrix} u_{sk}. \qquad \text{(Eq 4.96)}$$

The variables $\dot{\beta}_{pitch}$ and $\dot{\beta}_{yaw}$ are estimates of the LOS angle rates $\dot{\lambda}_{pitch}$ and $\dot{\lambda}_{yaw}$, and are available from the second and forth states of x_{sk}, respectively. The estimates of the LOS rate in pitch and yaw permits us to determine the missile command inputs u_{pitch} and u_{yaw}.

3. Guidance System

In this work, the guidance system dynamics are modeled as a single lag as seen in equation 4.66. This equation is repeated here.

$$\frac{a_m(s)}{u_m(s)} = \frac{1}{1+Ts}. \qquad \text{(Eq 4.97)}$$

For the 3-D missile/target engagement the guidance system generates missile commands in both planes, pitch and yaw. Hence:

$$\frac{a_{m\_pitch}(s)}{u_{pitch}(s)} = \frac{1}{1+Ts}, \qquad \text{(Eq 4.98)}$$

$$\frac{a_{m\_yaw}(s)}{u_{yaw}(s)} = \frac{1}{1+Ts}, \qquad \text{(Eq 4.99)}$$

where a_{m_pitch} and a_{m_yaw} are the pitch and yaw missile's accelerations; u_{pitch} and u_{yaw} are the pitch and yaw missile's acceleration commands. Figures 4.8 and 4.9, show the pitch and yaw signal flow graphs, for the missile guidance system. We chose the guidance system time constant T to be 1.0 second.

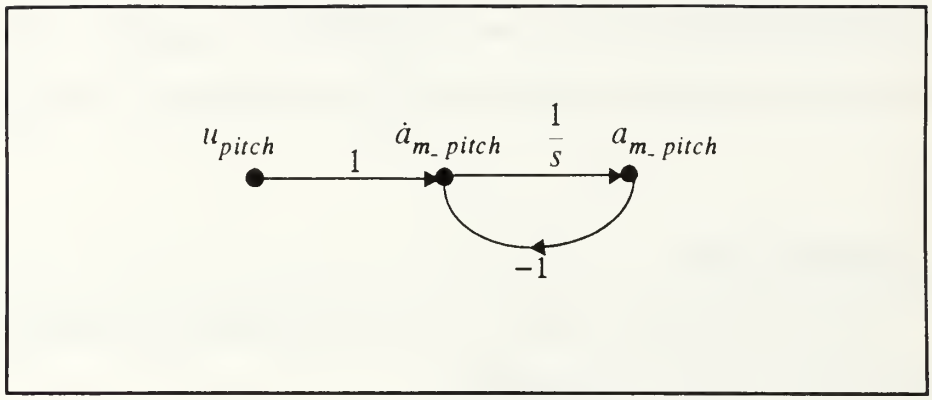

Figure 4.8 Guidance System Signal Flow Graph (Pitch)

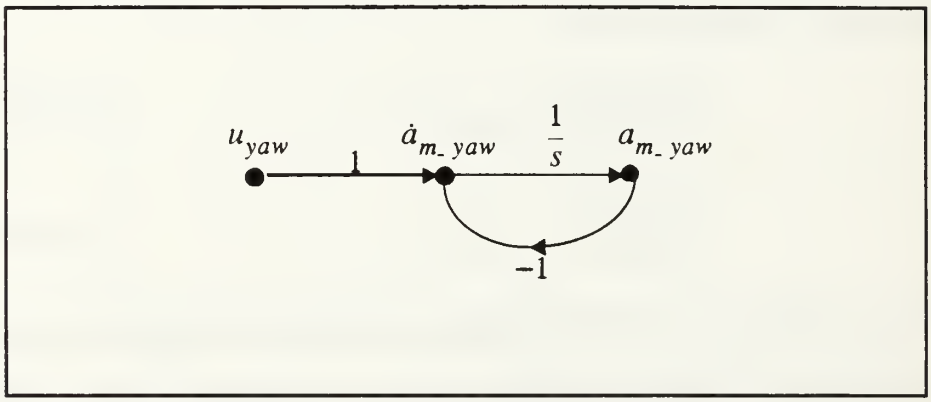

Figure 4.9 Guidance System Signal Flow Graph (Yaw)

From these diagrams, the state equation is easily obtained. Defining the guidance system state vector as:

$$x_{gs} = \begin{bmatrix} a_{m\_pitch} \\ a_{m\_yaw} \end{bmatrix}, \qquad \text{(Eq 4.100)}$$

and the guidance system input as:

$$u_{gs} = \begin{bmatrix} u_{pitch} \\ u_{yaw} \end{bmatrix}, \qquad \text{(Eq 4.101)}$$

the state equation becomes:

$$\dot{x}_{gs} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} x_{gs} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u_{gs}. \qquad \text{(Eq 4.102)}$$

4. Missile And Target Kinematics

The missile is controlled in three dimensional space by generating acceleration commands in two orthogonal planes. These planes are the pitch and yaw planes. The acceleration commands in pitch and yaw are issued perpendicular to the respective lines of sight. The magnitude of the acceleration commands depends on the selected guidance law. From equation 4.84, the pitch and yaw missile acceleration commands may be expressed, in its general form, as:

$$u_{pitch} = NV_{c\_pitch}\dot{\lambda}_{pitch} + f_{pitch}(a_m, a_t, t_{go}, T), \qquad \text{(Eq 4.103)}$$

$$u_{yaw} = NV_{c\_yaw}\dot{\lambda}_{yaw} + f_{yaw}(a_m, a_t, t_{go}, T). \qquad \text{(Eq 4.104)}$$

V_{c_pitch} and V_{c_yaw} are the relative speeds, between the target and the missile along the pitch and yaw line of sights. The functions $f_{pitch}(a_m, a_t, t_{go}, T)$ and $f_{yaw}(a_m, a_t, t_{go}, T)$ are the augmented PROPNAV or optimal guidance extra terms in pitch and yaw, respectively.

In order to track the missile's 3-D coordinates (x_m, y_m, z_m), the missile command accelerations in pitch and yaw, are broken down into cartesian coordinate system components.

Figure 4.10, shows the decomposition of the pitch acceleration command in its components. From this figure the following relationships are derived:

$$a_{mx\_pitch} = -(a_{m\_pitch}\sin\lambda_{pitch})\cos\lambda_{yaw}, \qquad \text{(Eq 4.105)}$$

$$a_{mx\_pitch} = -(a_{m\_pitch}\sin\lambda_{pitch})\sin\lambda_{yaw}, \qquad \text{(Eq 4.106)}$$

$$a_{mz\_pitch} = a_{m\_pitch}\cos\lambda_{pitch}. \qquad \text{(Eq 4.107)}$$

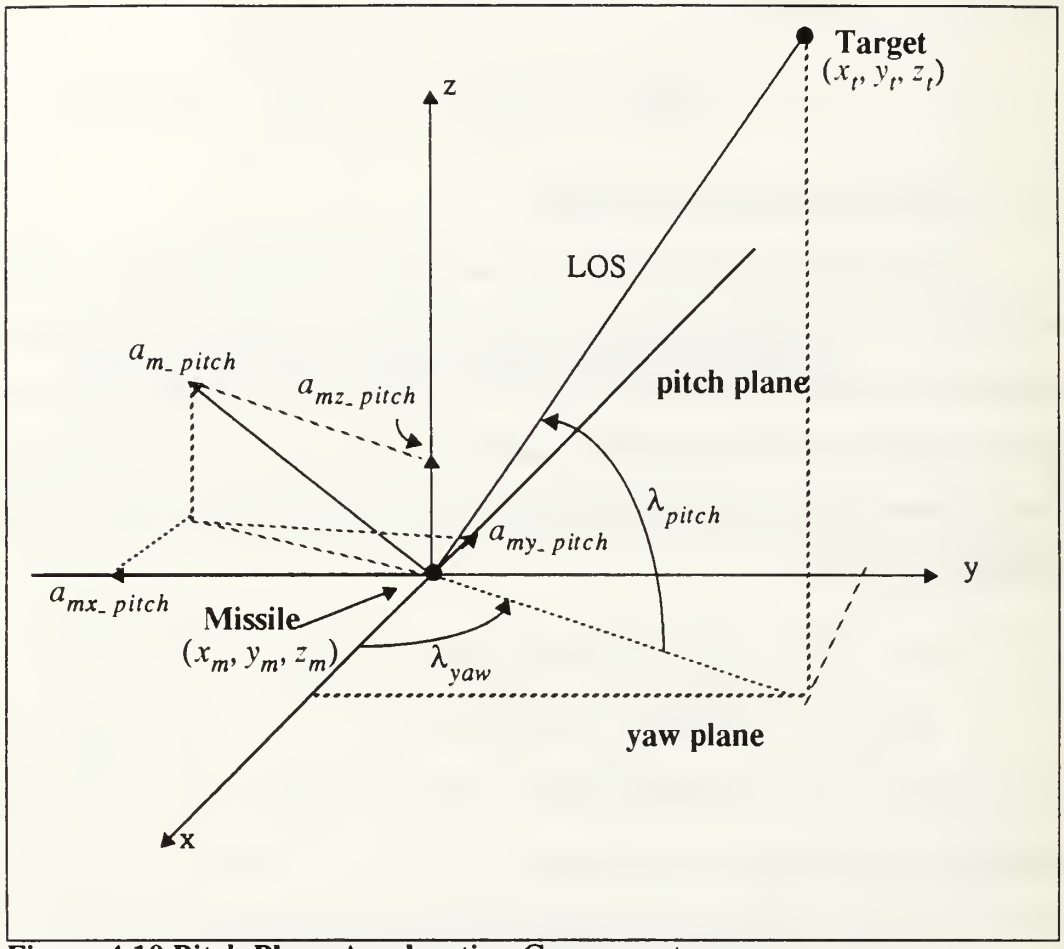

Figure 4.10 Pitch Plane Acceleration Components

Figure 4.11 shows the decomposition of the yaw plane acceleration command in to its x and y components. From the figure the following relationships are obtained:

$$a_{mx\_yaw} = -a_{m\_yaw}\sin\lambda_{yaw}, \qquad \text{(Eq 4.108)}$$

$$a_{my\_yaw} = a_{m\_yaw}\cos\lambda_{yaw}. \qquad \text{(Eq 4.109)}$$

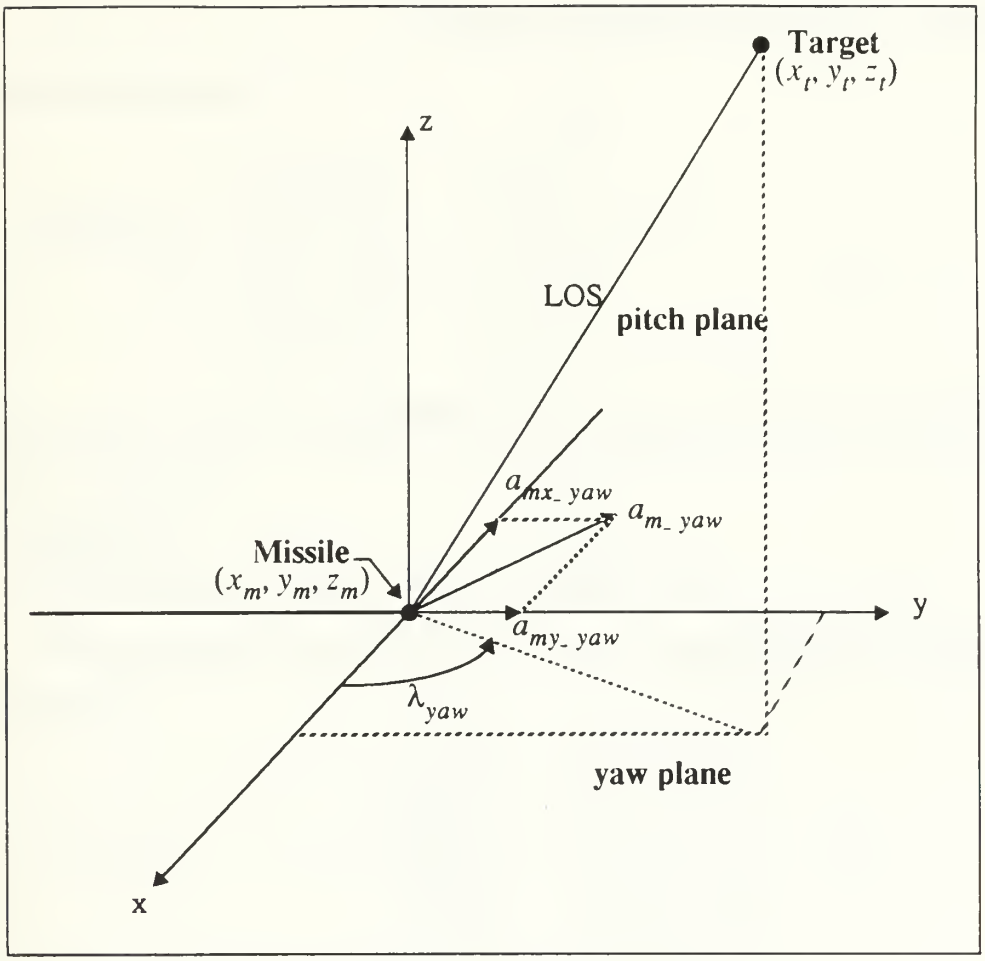

Figure 4.11 Yaw Plane Acceleration Components

To find the overall missile's acceleration components along the three cartesian axis, we use the results of the equations 4.105 through 4.109:

$$a_{mx} = a_{mx\_pitch} + a_{mx\_yaw}, \qquad \text{(Eq 4.110)}$$

$$a_{my} = a_{my\_pitch} + a_{my\_yaw}, \qquad \text{(Eq 4.111)}$$

$$a_{mz} = a_{mz\_pitch}. \qquad \text{(Eq 4.112)}$$

The missile's tridimensional movement is determined by these three acceleration components. Defining the missile state vector as:

$$X_m = \begin{bmatrix} x_m \\ \dot{x}_m \\ y_m \\ \dot{y}_m \\ z_m \\ \dot{z}_m \end{bmatrix}, \qquad \text{(Eq 4.113)}$$

and the input as the missile's acceleration command:

$$a_m = \begin{bmatrix} a_{mx} \\ a_{my} \\ a_{mz} \end{bmatrix}, \qquad \text{(Eq 4.114)}$$

the missile state equation is:

$$\dot{X}_m = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} X_m + \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} a_m. \qquad \text{(Eq 4.115)}$$

As we have seen in Chapter III, the missile when equipped with a camera and a seeker is able to estimate the target's 3-D acceleration components. This information may be used to improve the missile guidance towards the target. Defining the target state vector as:

$$X_t = \begin{bmatrix} x_t \\ \dot{x}_t \\ y_t \\ \dot{y}_t \\ z_t \\ \dot{z}_t \end{bmatrix}, \qquad \text{(Eq 4.116)}$$

and the target's acceleration as:

$$a_t = \begin{bmatrix} a_{tx} \\ a_{ty} \\ a_{tz} \end{bmatrix}, \qquad \text{(Eq 4.117)}$$

the target state equation is:

$$\dot{X}_t = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} X_t + \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} a_t; \qquad \text{(Eq 4.118)}$$

the tridimensional target acceleration a_t may be estimated using the missile's image processing capabilities;

5. Pitch And Yaw Closing Velocities. Determination Of Time To Go

Figure 4.12 shows the decomposition of the missile and target absolute velocities. From this figure:

$$V_m = \begin{bmatrix} V_{mx} \\ V_{my} \\ V_{mz} \end{bmatrix}, \qquad \text{(Eq 4.119)}$$

$$V_t = \begin{bmatrix} V_{tx} \\ V_{ty} \\ V_{tz} \end{bmatrix}. \qquad \text{(Eq 4.120)}$$

Figures 4.13 and 4.14 show the projection of the missile's velocity vector over the pitch and yaw planes.

These figures permit us to compute the missile's and target's velocity components, over the pitch and yaw planes:

$$V_{m\_pitch} = V_m \cos(\gamma_{m\_yaw} - \lambda_{yaw}), \qquad \text{(Eq 4.121)}$$

$$V_{m\_yaw} = V_m \cos\gamma_{m\_ver}, \qquad \text{(Eq 4.122)}$$

and

$$V_{t\_pitch} = V_t \cos(\gamma_{t\_yaw} - \lambda_{yaw}), \quad \text{(Eq 4.123)}$$

$$V_{t\_yaw} = V_t \cos\gamma_{t\_ver}. \quad \text{(Eq 4.124)}$$

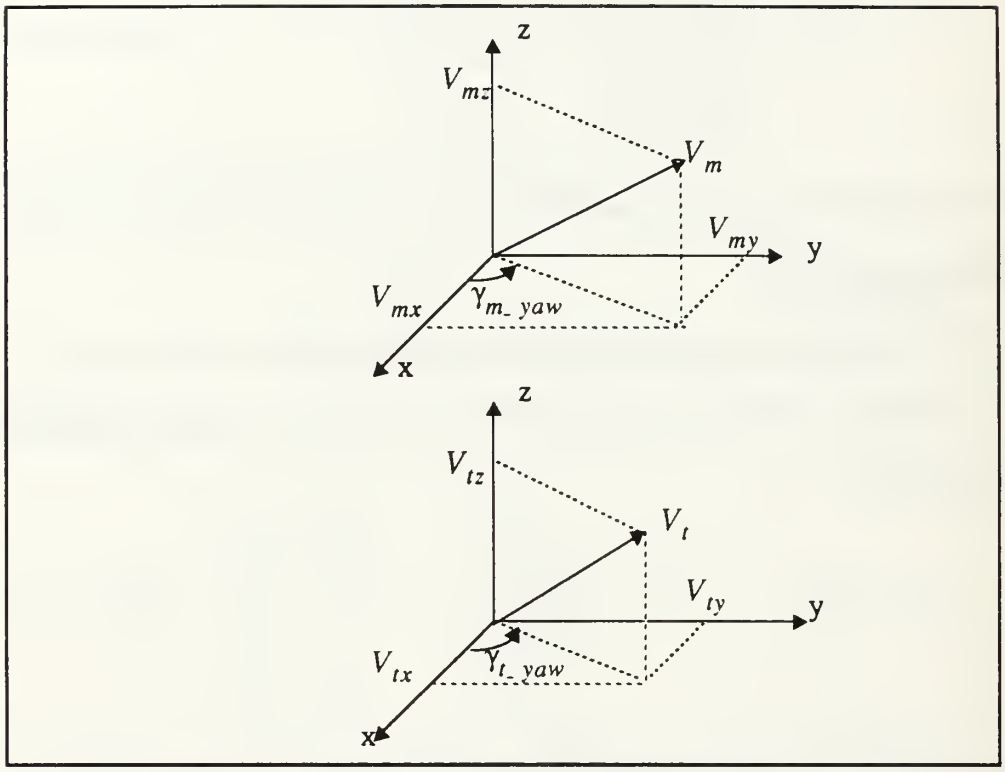

Figure 4.12 Missile And Target Velocity Components

The pitch closing velocity V_{c_pitch} is found by projecting the missile's and target's pitch plane velocities along the LOS in pitch (see Figure 4.4). Then:

(Eq 4.125)

$$V_{c\_pitch} = V_{m\_pitch} \cos(\lambda_{pitch} - \gamma_{m\_pitch}) - V_{t\_pitch} \cos(\lambda_{pitch} - \gamma_{t\_pitch}).$$

Similarly, the yaw plane closing velocity, is obtained as:

(Eq 4.126)

$$V_{c\_yaw} = V_{m\_yaw}\cos(\gamma_{m\_yaw} - \lambda_{yaw}) - V_{t\_yaw}\cos(\gamma_{t\_yaw} - \lambda_{yaw}) \cdot$$

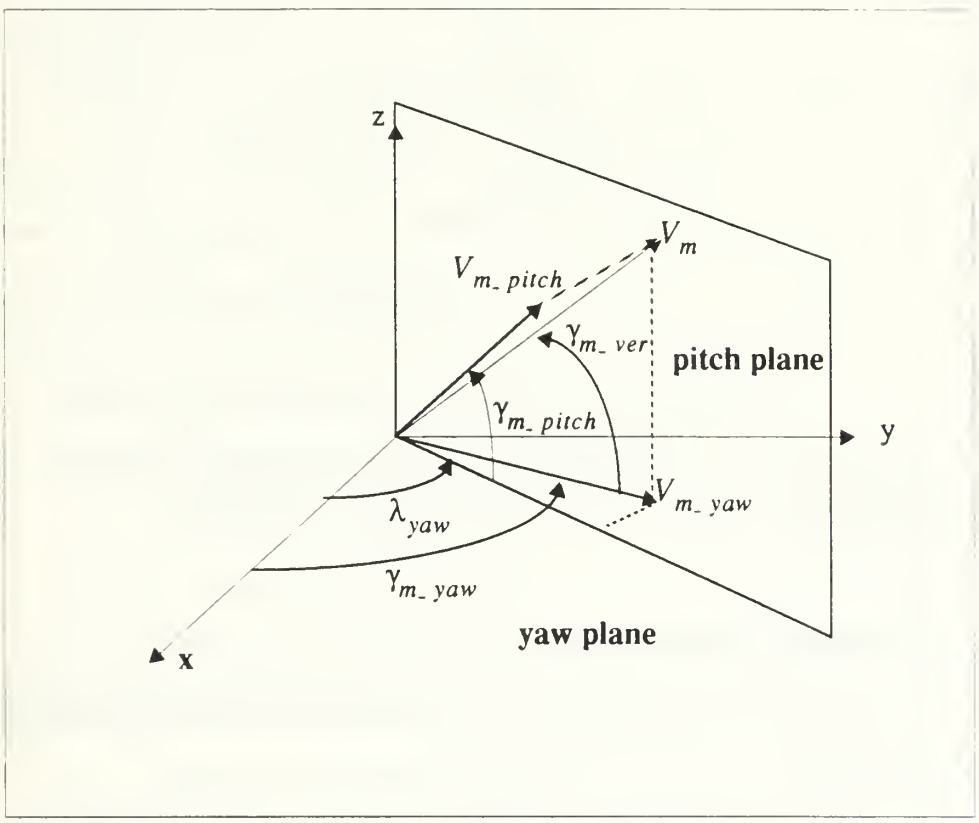

Figure 4.13 Missile's Pitch And Yaw Velocity Components

The time to go until interception may be computed from:

$$t_{go} = \frac{r}{V_{c\_pitch}}, \qquad \text{(Eq 4.127)}$$

where r is the missile/target relative distance along the pitch LOS (see Figure 4.4).

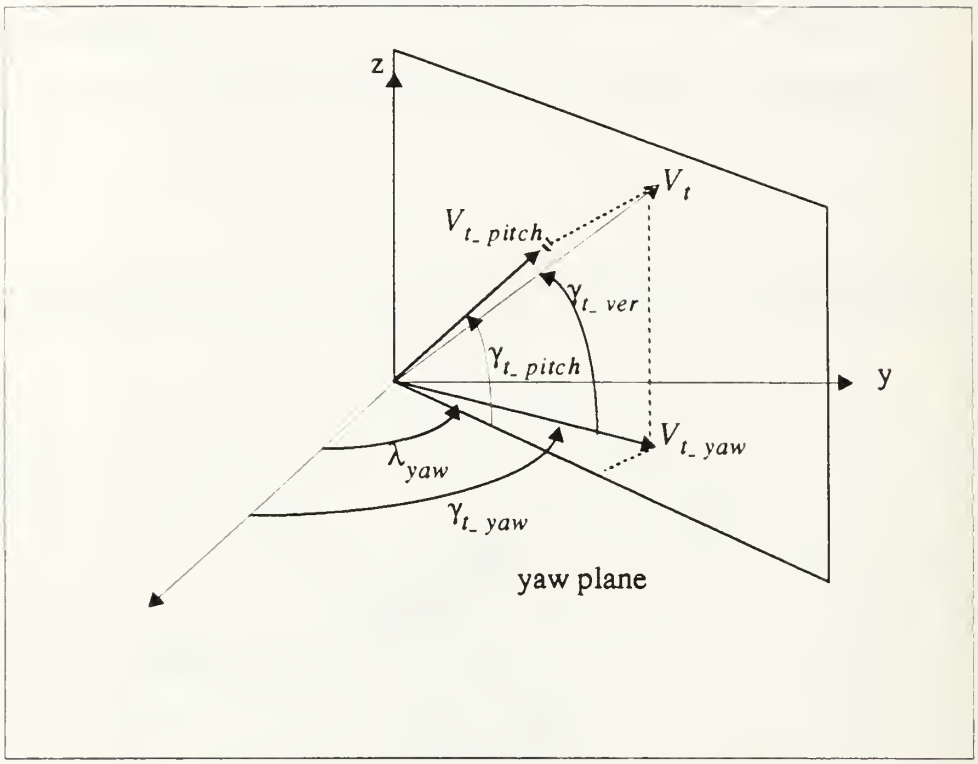

Figure 4.14 Target's Pitch And Yaw Velocity Components

6. Proportional Navigation

The PROPNAV 3-D simulation, in Appendix A, uses missile commands in pitch and yaw of the form:

$$u_{pitch} = NV_{c\_pitch}\dot{\lambda}_{pitch}, \qquad \text{(Eq 4.128)}$$

$$u_{yaw} = NV_{c\_yaw}\dot{\lambda}_{yaw} \qquad \text{(Eq 4.129)}$$

where N is a constant.

7. Augmented Proportional Navigation

The augmented PROPNAV 3-D simulation, in Appendix B, uses missile commands in pitch and yaw of the form:

$$u_{pitch} = NV_{c\_pitch}\dot{\lambda}_{pitch} + 0.5Na_{t\_pitch}, \qquad \text{(Eq 4.130)}$$

$$u_{yaw} = NV_{c\_yaw}\dot{\lambda}_{yaw} + 0.5Na_{t\_yaw} \qquad \text{(Eq 4.131)}$$

where a_{t_pitch} and a_{t_yaw} are the components of the target 's acceleration normal to the pitch and yaw line of sights and N is a constant.

8. Optimal Guidance

The optimal guidance 3-D simulation, in Appendix C, uses missile commands in pitch and yaw of the form:

$$u_{pitch} = NV_{c\_pitch}\dot{\lambda}_{pitch} + \frac{N}{k^2}(e^{-k} + k - 1)\, a_{m\_pitch} + \frac{N}{2}a_{t\_pitch}, \qquad \text{(Eq 4.132)}$$

$$u_{yaw} = NV_{c\_yaw}\dot{\lambda}_{yaw} + \frac{N}{k^2}(e^{-k} + k - 1)\, a_{m\_yaw} + \frac{N}{2}a_{t\_yaw} \qquad \text{(Eq 4.133)}$$

where k and N are given by equation 4.75.

9. Discrete-Time Simulation Using State Space Methods

The general continuous-time state equations are:

$$\dot{x}(t) = Ax(t) + Bu(t), \qquad \text{(Eq 4.134)}$$

$$y(t) = Cx(t) + Du(t) \qquad \text{(Eq 4.135)}$$

where $x(t)$ is the state vector and $y(t)$ is the output vector. This system is simulated by iterating the discrete-time state equations:

$$x(n+1) = \Phi x(n) + \Gamma u(n), \qquad \text{(Eq 4.136)}$$

$$y(n) = Cx(n) + Du(n). \qquad \text{(Eq 4.137)}$$

where:

$$\Phi = e^{AT_s}; \quad T_s = \text{sampling time}; \qquad \text{(Eq 4.138)}$$

$$\Delta = \int_0^{T_s} e^{At} B \; dt. \qquad \text{(Eq 4.139)}$$

The missile/target engagement scenario is simulated using the MATLAB software package. The discrete-time state equations used in the simulation are defined for the seeker, guidance system, missile, and target dynamics:

$$x_{sk}(n+1) = \Phi_{sk} x_{sk}(n) + \Gamma_{sk} u_{sk}(n); \quad \text{(Eq 4.140)}$$

$$x_{gs}(n+1) = \Phi_{gs} x_{gs}(n) + \Gamma_{gs} u_{gs}(n); \quad \text{(Eq 4.141)}$$

$$x_{m}(n+1) = \Phi_{m} x_{m}(n) + \Gamma_{m} u_{m}(n); \quad \text{(Eq 4.142)}$$

$$x_{t}(n+1) = \Phi_{t} x_{t}(n) + \Gamma_{t} u_{t}(n). \quad \text{(Eq 4.143)}$$

V. SIMULATION RESULTS

A. GENERAL

This chapter presents the results of the computer simulations for a missile equipped with a radar and a camera. The missile/target 3-D engagement was simulated using three control laws:

1. Proportional navigation,
2. Augmented proportional navigation,
3. Optimal guidance.

The simulation was conducted for two different target maneuvers:

1. A 3-D constant target acceleration,
2. A 3-D varying target acceleration.

The following assumptions are made throughout:

1. We assume that the simulation's initial conditions are defined when the missile enters into the terminal phase of the flight (about 10 seconds before impact).
2. The PROPNAV and APROPNAV effective navigation constant is 3 ($N = 3$),
3. The missile is limited to 25 g's accelerations in pitch and yaw,
4. The instantaneous target acceleration is available from previous image processing. No delays are assumed in this process.
5. The missile and target speeds are limited to 3500 and 2000 feet/sec, respectively,
6. The missile is in a collision triangle with the target on entering into the terminal phase of flight.
7. The target may start its evasion maneuver at any time (initial time),
8. The target is limited to a maximum of 12 g's,
9. The acceleration due to gravity is ignored,
10. The sampling time is 0.01 seconds.

B. ENGAGEMENT SCENARIOS. RESULTS

1. Scenario #1 (Constant Target Acceleration)

The initial missile/target geometry (when the missile enters into the terminal phase of flight) is shown in Figure 5.1.

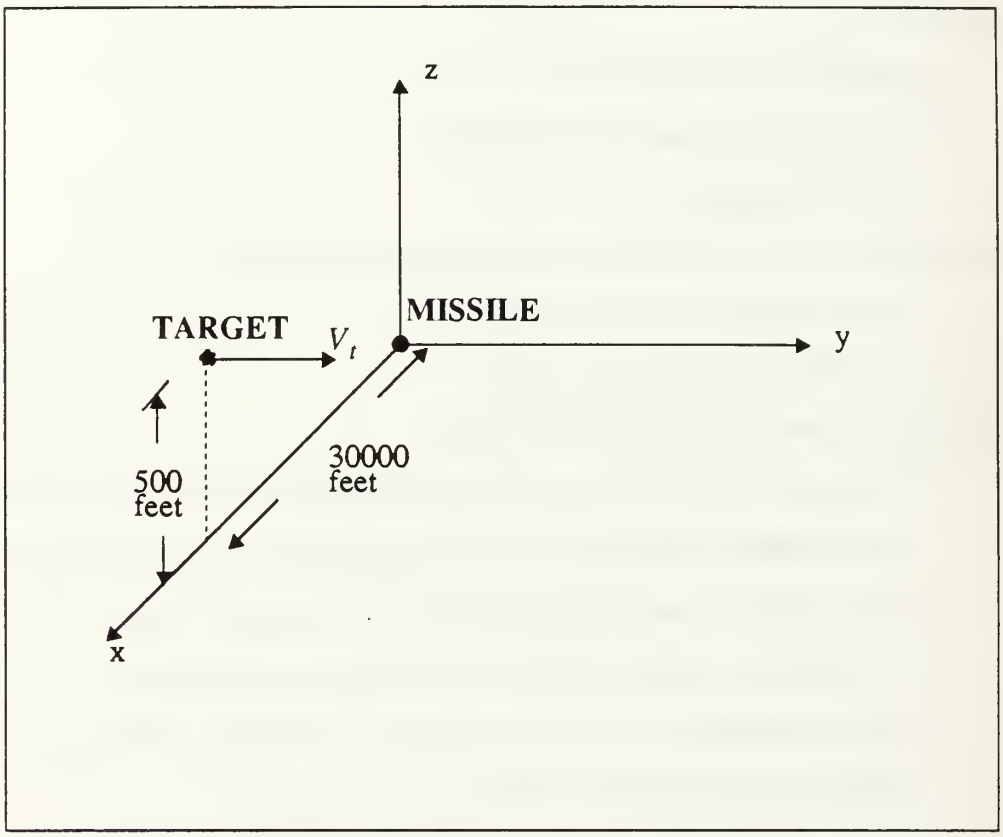

Figure 5.1 Initial Missile/Target Geometry

The engagement initial conditions are (distances are in feet and speeds in feet/ sec):

$$X_m(0) = \begin{bmatrix} x_m(0) \\ \dot{x}_m(0) \\ y_m(0) \\ \dot{y}_m(0) \\ z_m(0) \\ \dot{z}_m(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 2828 \\ 0 \\ 1000 \\ 0 \\ 47.1339 \end{bmatrix}, \qquad \text{(Eq 5.1)}$$

$$X_t(0) = \begin{bmatrix} x_t(0) \\ \dot{x}_t(0) \\ y_t(0) \\ \dot{y}_t(0) \\ z_t(0) \\ \dot{z}_t(0) \end{bmatrix} = \begin{bmatrix} 30000 \\ 0 \\ 0 \\ 1000 \\ 500 \\ 0 \end{bmatrix}. \qquad \text{(Eq 5.2)}$$

The target's evasive maneuver is constant (the accelerations are in feet/sec^2)

$$\begin{bmatrix} \ddot{x}_t \\ \ddot{y}_t \\ \ddot{z}_t \end{bmatrix} = \begin{bmatrix} 3 \times g \\ 1 \times g \\ 2 \times g \end{bmatrix}, \qquad \text{(Eq 5.3)}$$

where the acceleration of gravity is: $g = 32.2$ feet/sec^2. Figures 5.2 to 5.22 display the results of the three dimensional simulation for the constant target evasive maneuver. Figures 5.2 to 5.8 relate to the proportional navigation (PROPNAV) guidance law. Figures 5.9 to 5.15 relate to the augmented proportional (APROPNAV) guidance law. Figures 5.16 to 5.22 relate to the optimal guidance law. Figures 5.4 to 5.8, 5.11 to 5.15 and 5.18 to 5.22 display the results assuming that the target starts its maneuver 6 seconds after the missile entered into the terminal phase of flight.

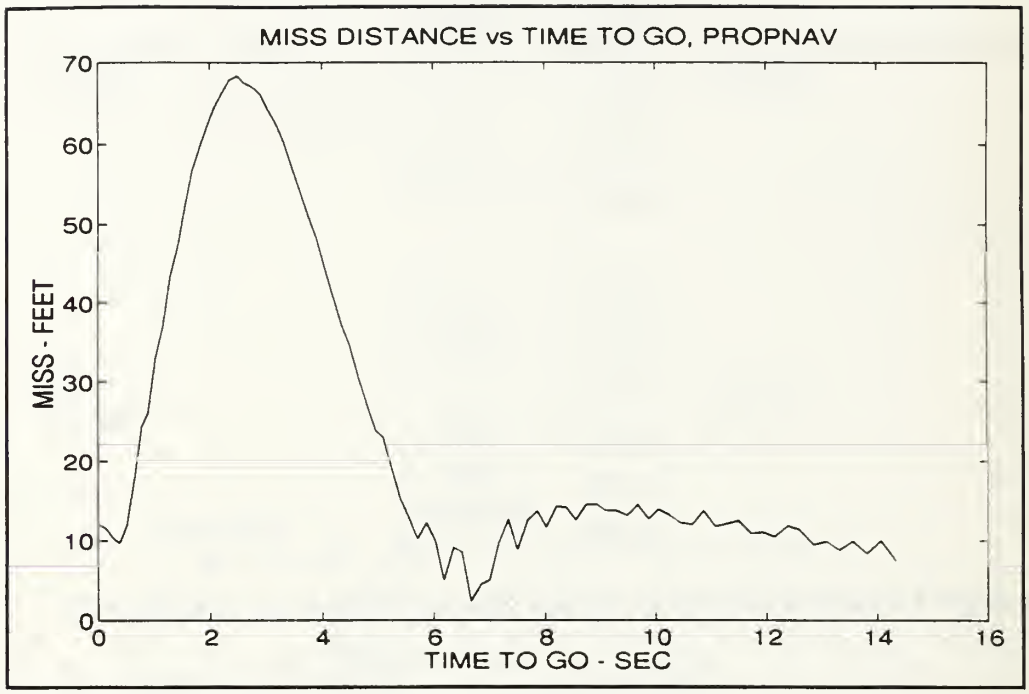

Figure 5.2 Miss Distance vs.Time To Go (PROPNAV)

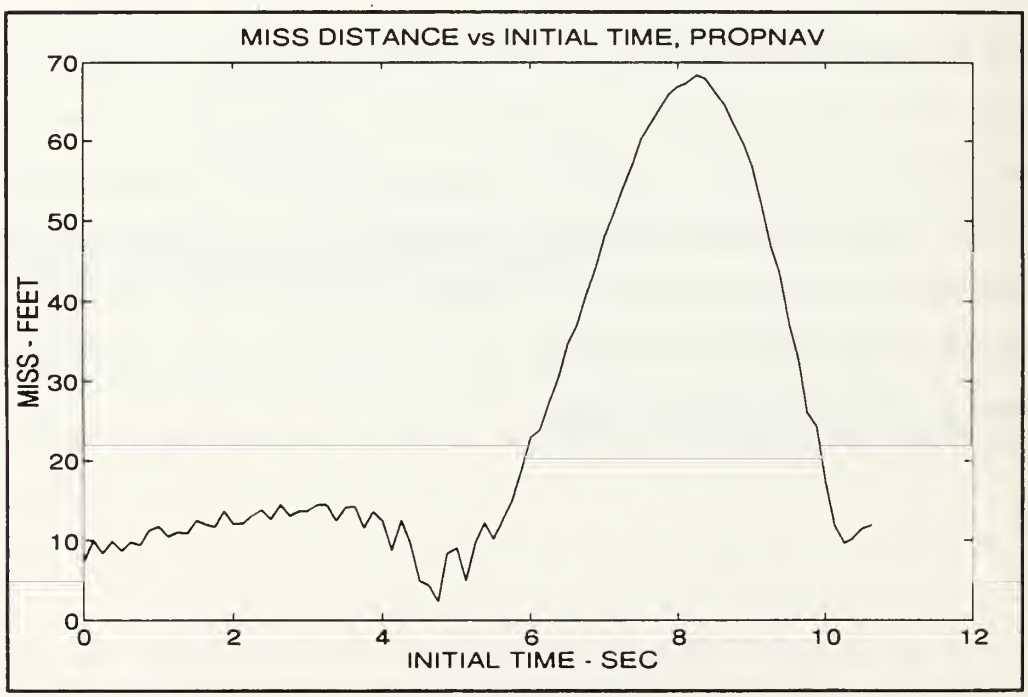

Figure 5.3 Miss Distance vs. Initial Time (PROPNAV)

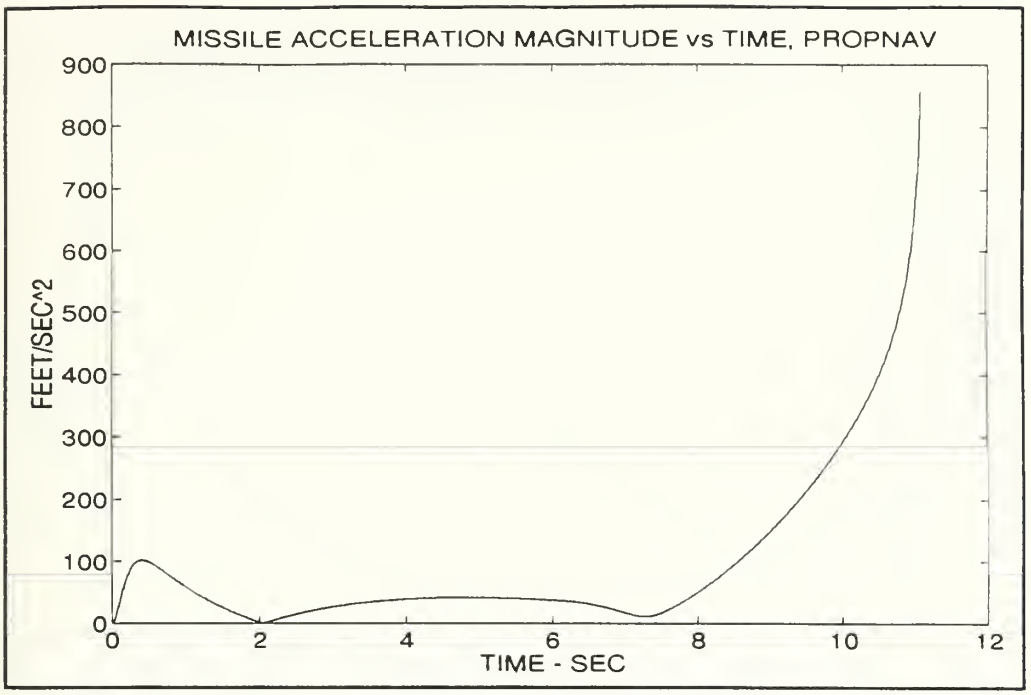

Figure 5.4 Missile Acceleration Magnitude (PROPNAV)

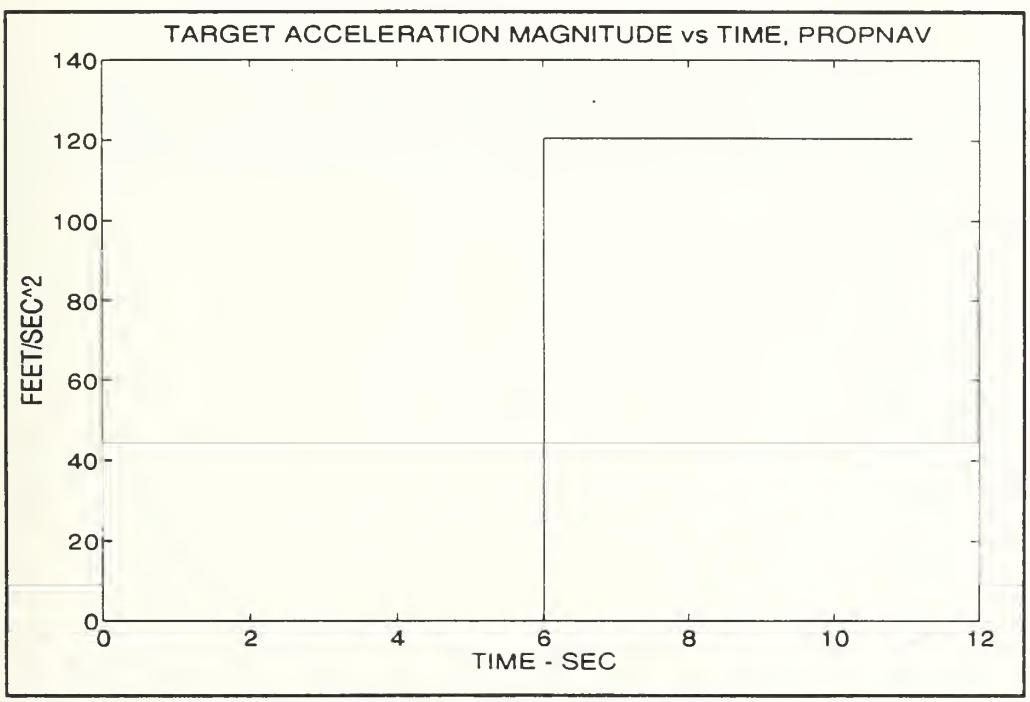

Figure 5.5 Target Acceleration Magnitude (PROPNAV)

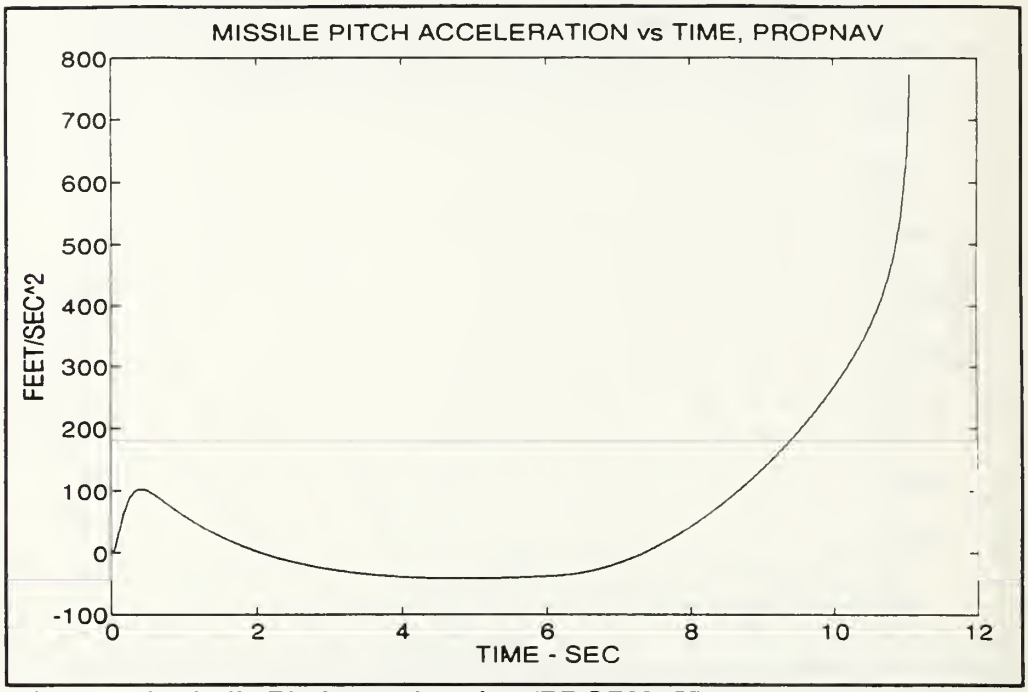

Figure 5.6 Missile Pitch Acceleration (PROPNAV)

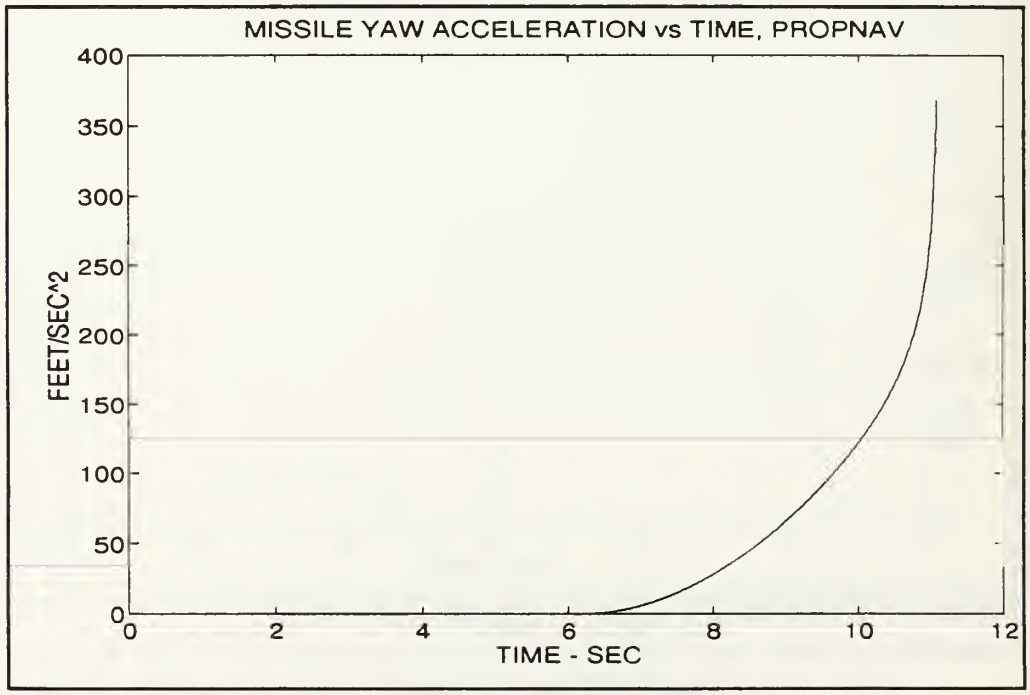

Figure 5.7 Missile Yaw Acceleration (PROPNAV)

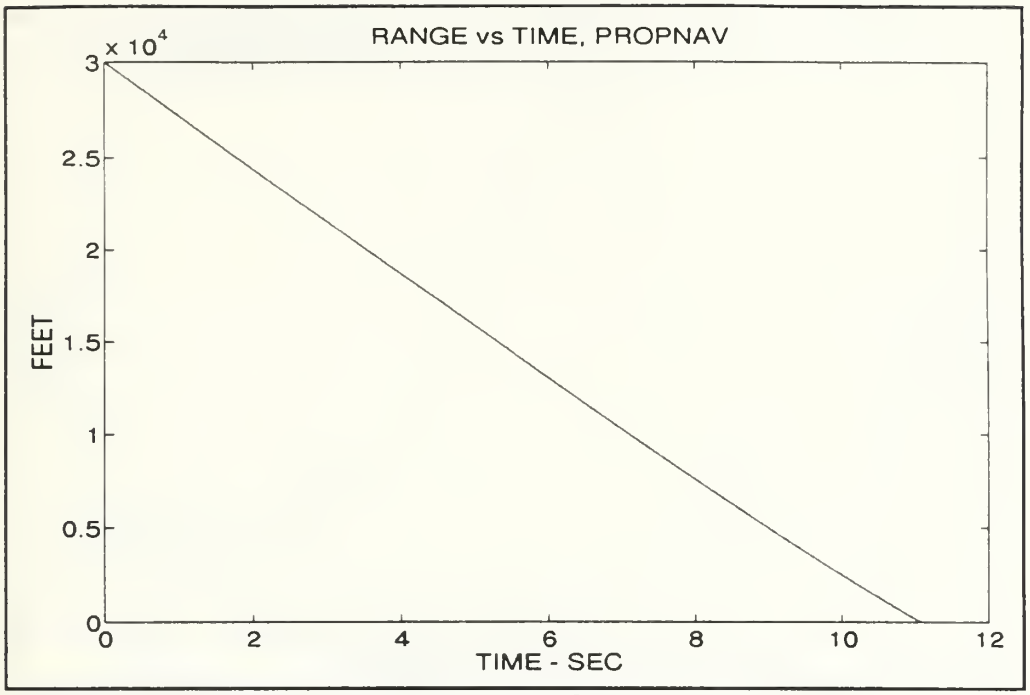

Figure 5.8 Missile To Target Range (PROPNAV)

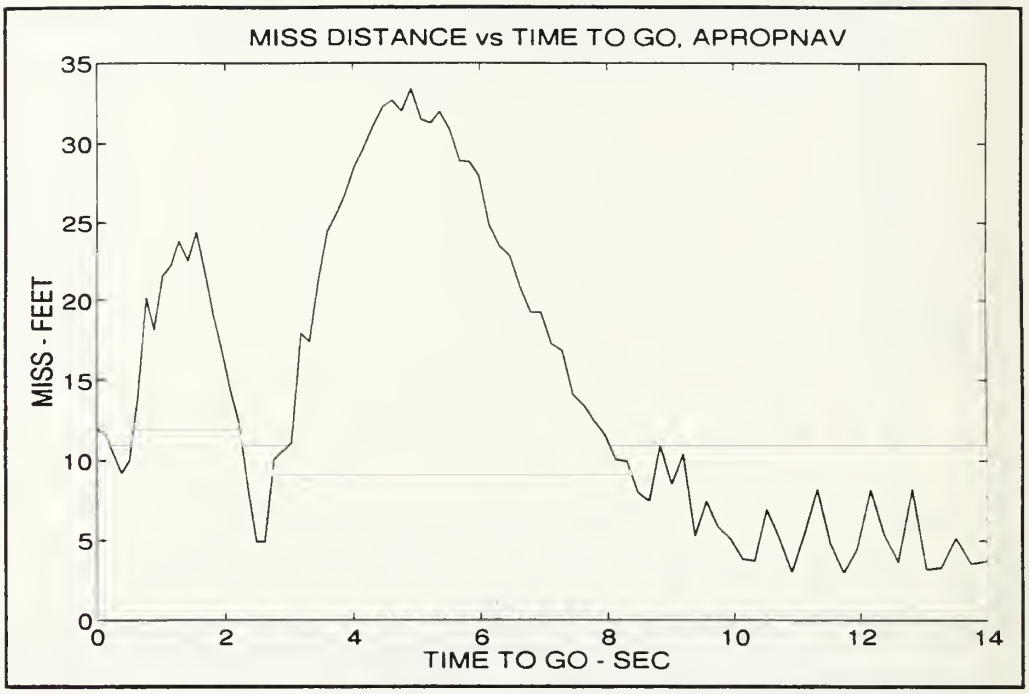

Figure 5.9 Miss Distance vs. Time To Go (APROPNAV)

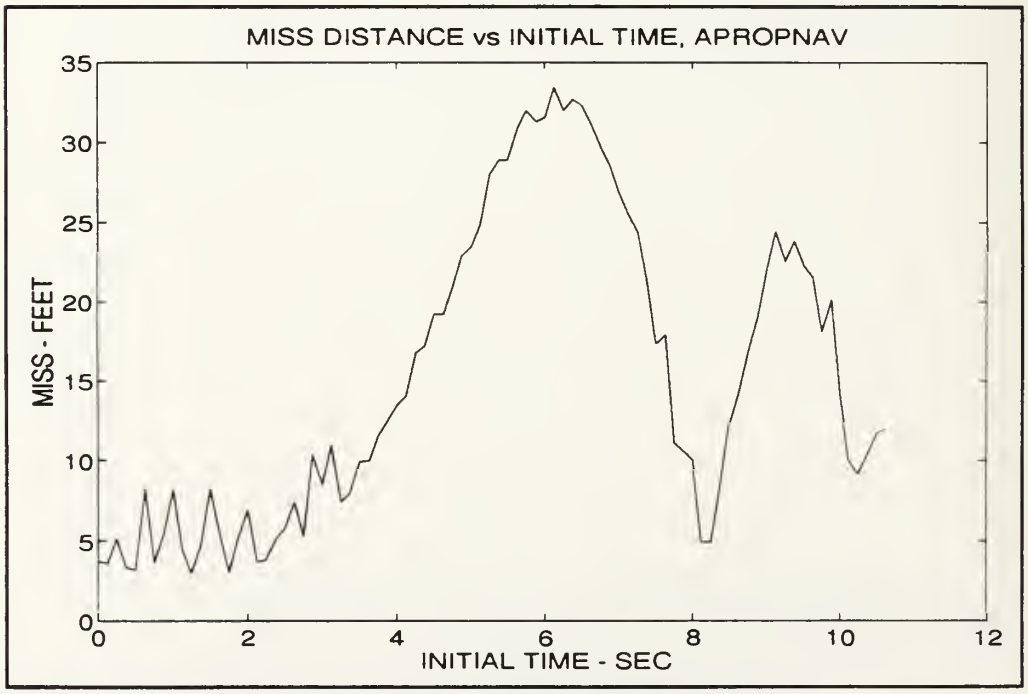

Figure 5.10 Miss Distance vs. Initial Time (APROPNAV)

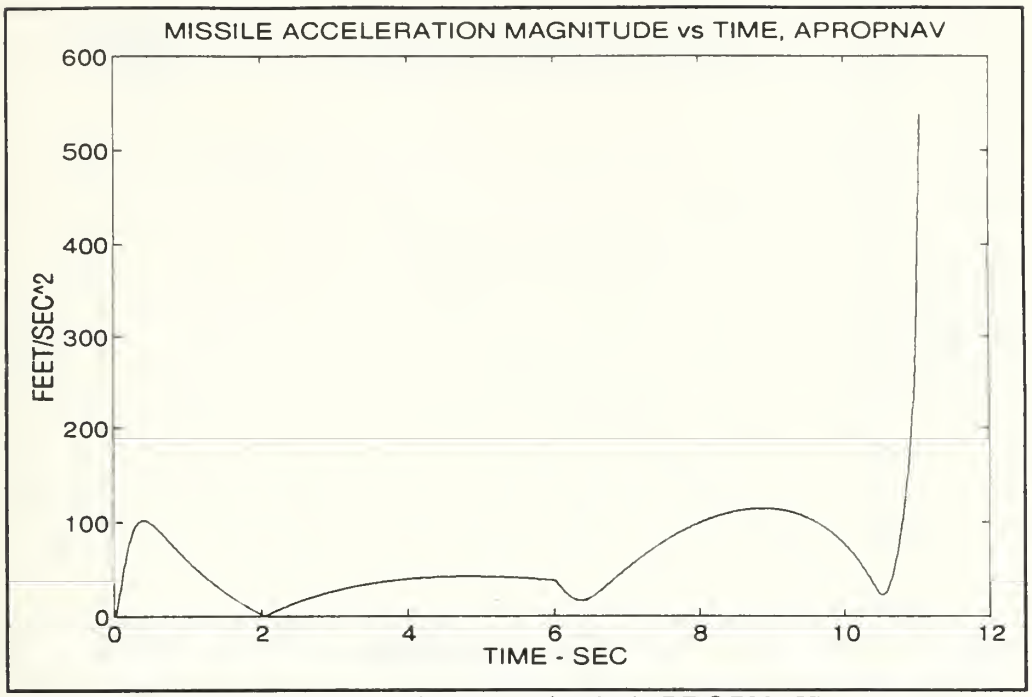

Figure 5.11 Missile Acceleration Magnitude (APROPNAV)

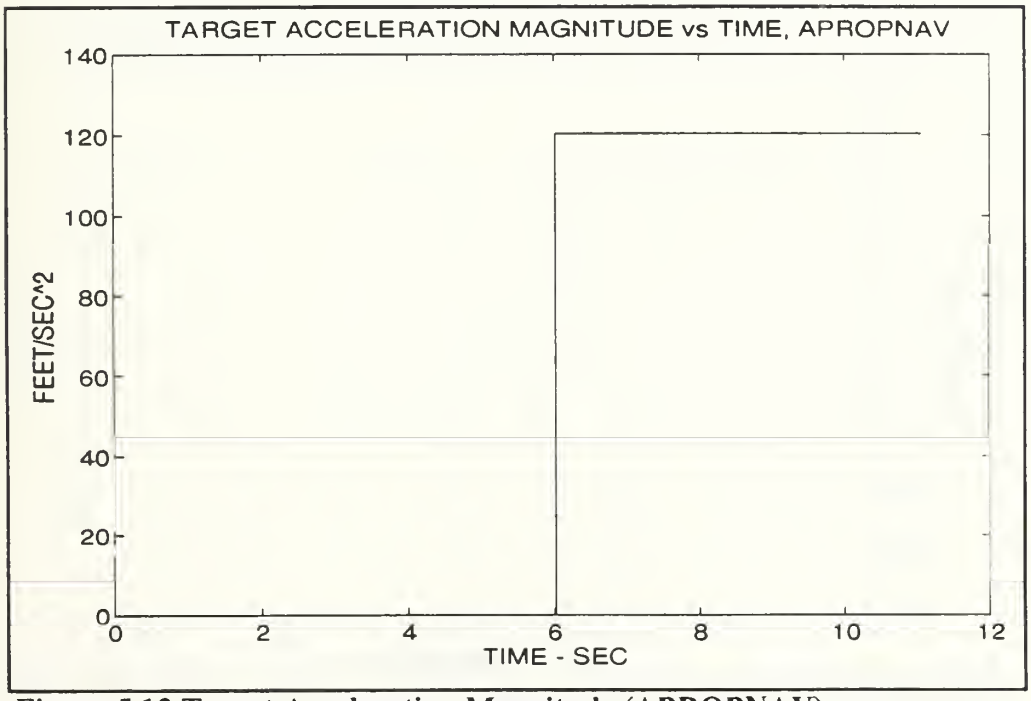

Figure 5.12 Target Acceleration Magnitude (APROPNAV)

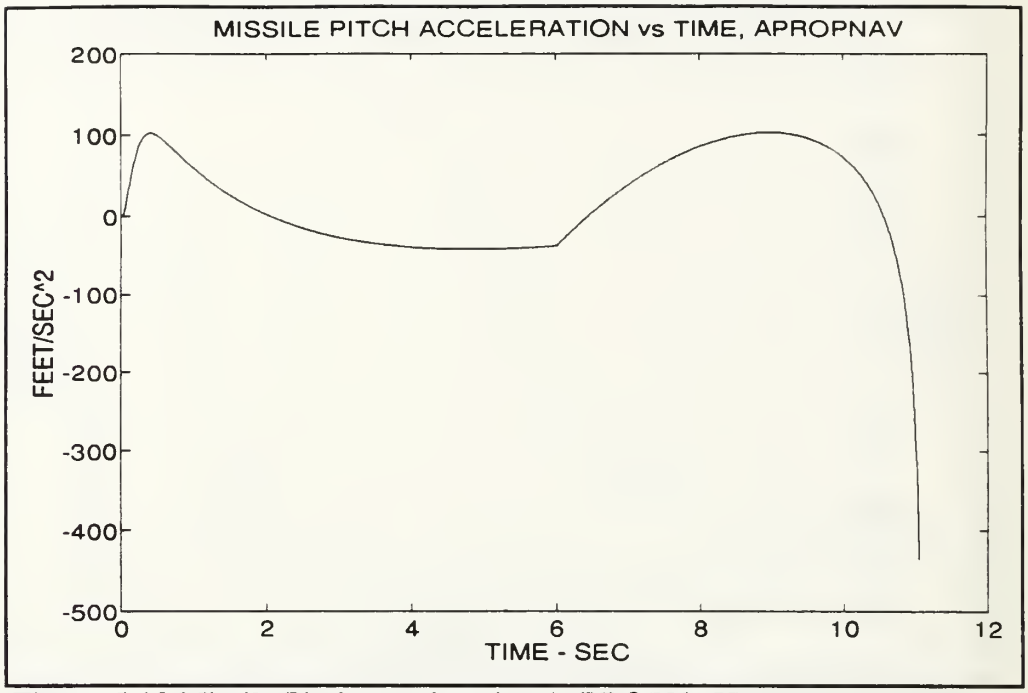

Figure 5.13 Missile Pitch Acceleration (APROPNAV)

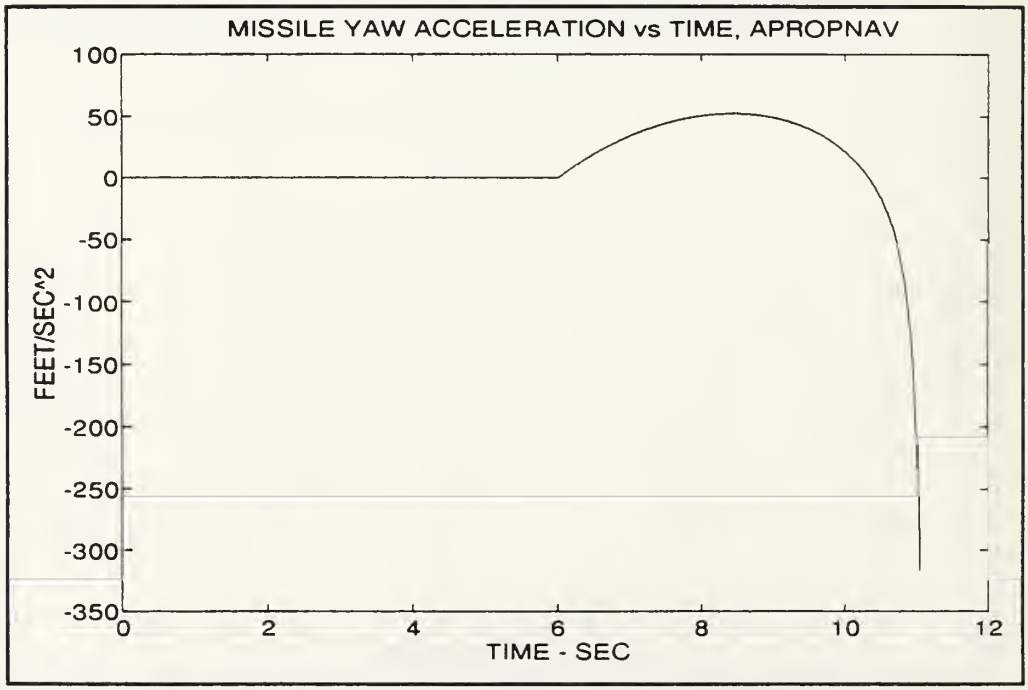

Figure 5.14 Missile Yaw Acceleration (APROPNAV)

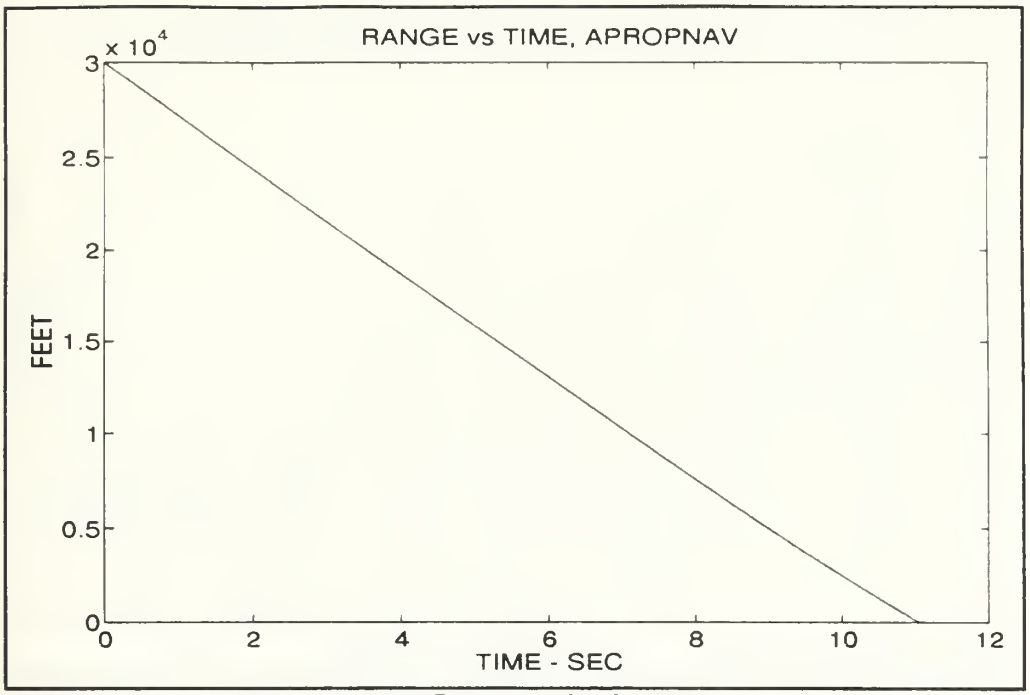

Figure 5.15 Missile To Target Range (APROPNAV)

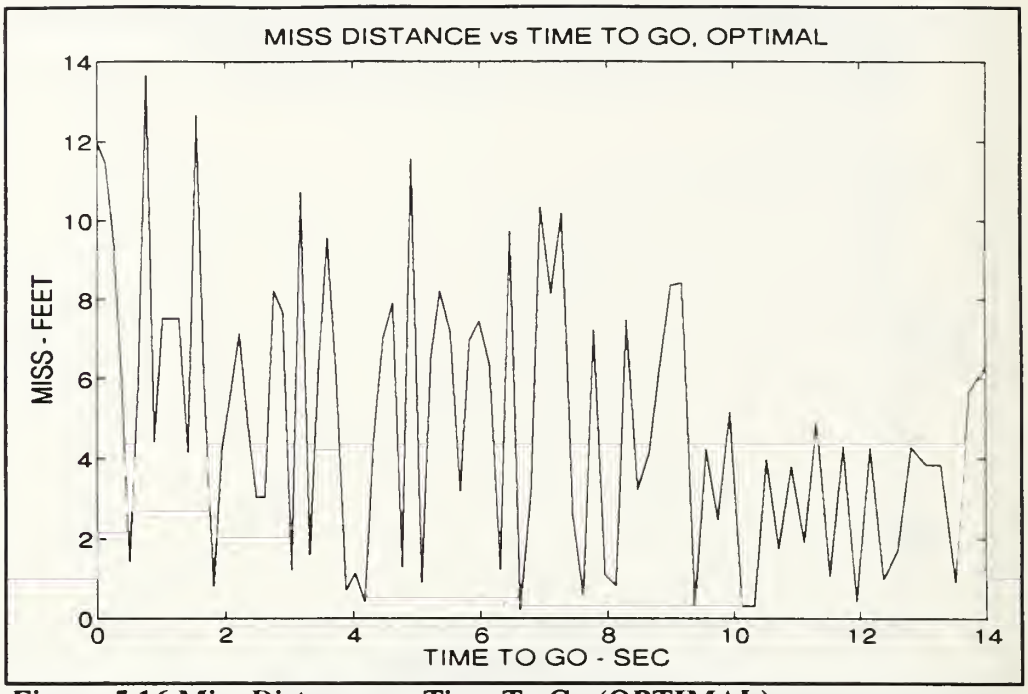

Figure 5.16 Miss Distance vs. Time To Go (OPTIMAL)

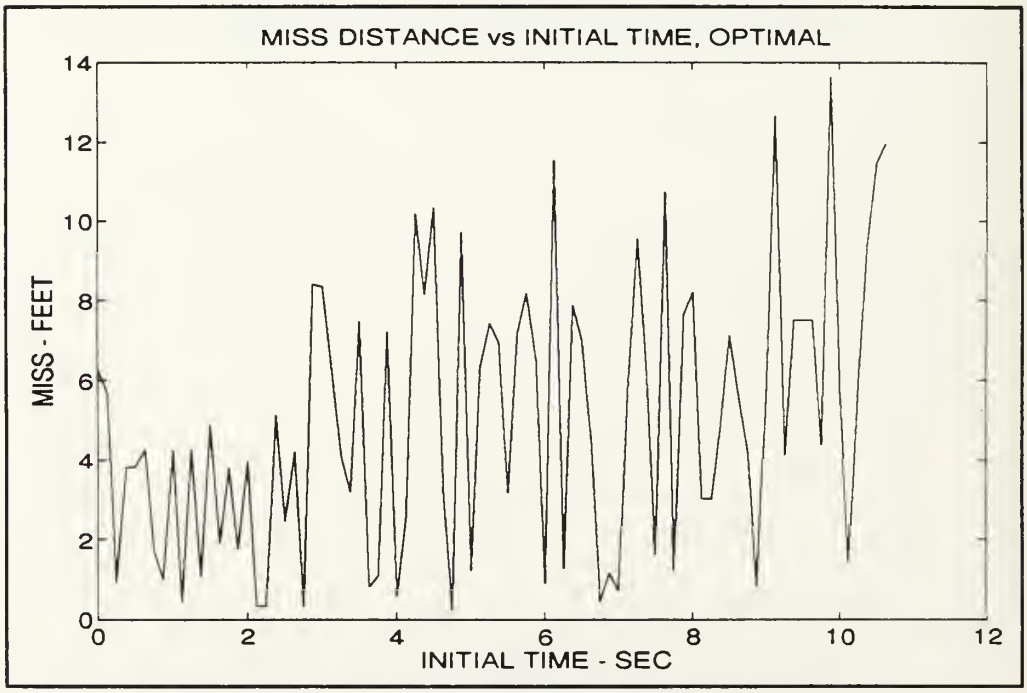

Figure 5.17 Miss Distance vs. Initial Time (OPTIMAL)

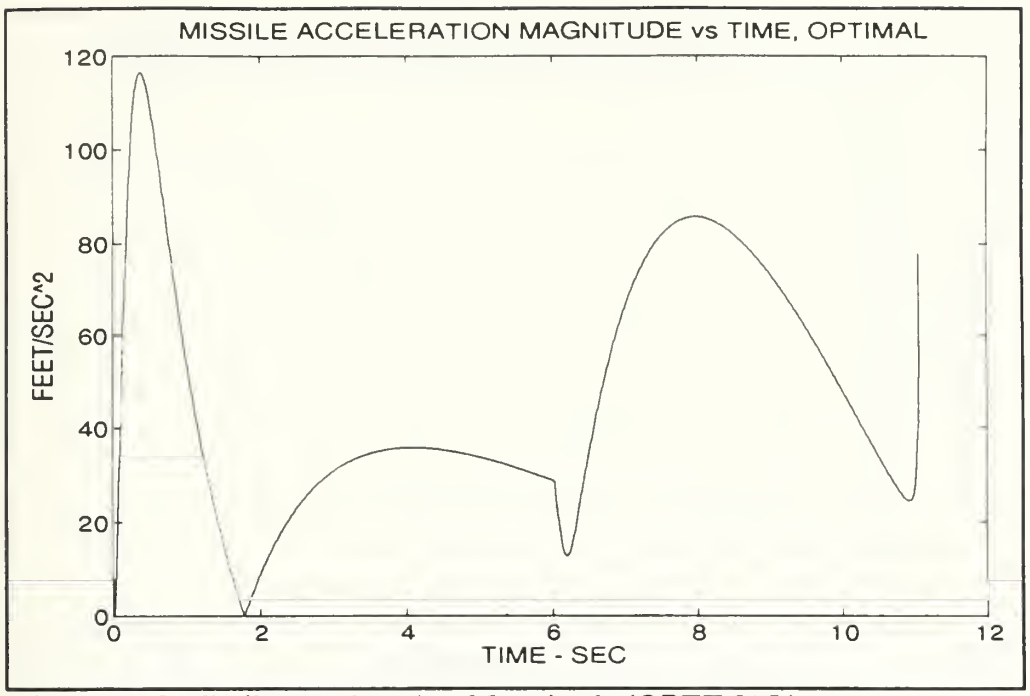

Figure 5.18 Missile Acceleration Magnitude (OPTIMAL)

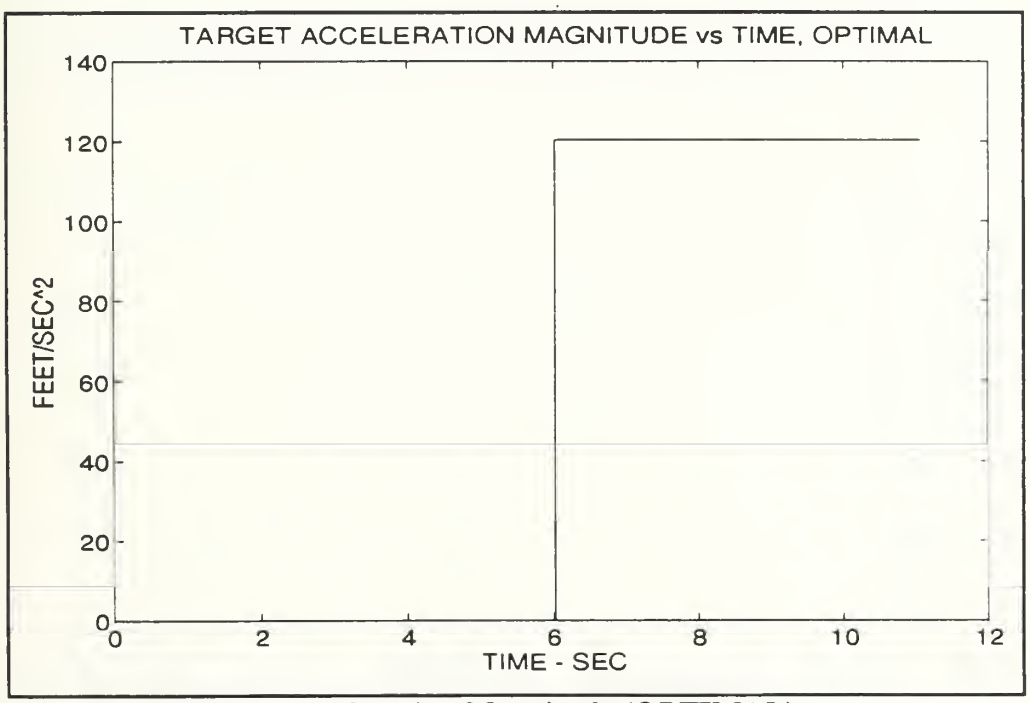

Figure 5.19 Target Acceleration Magnitude (OPTIMAL)

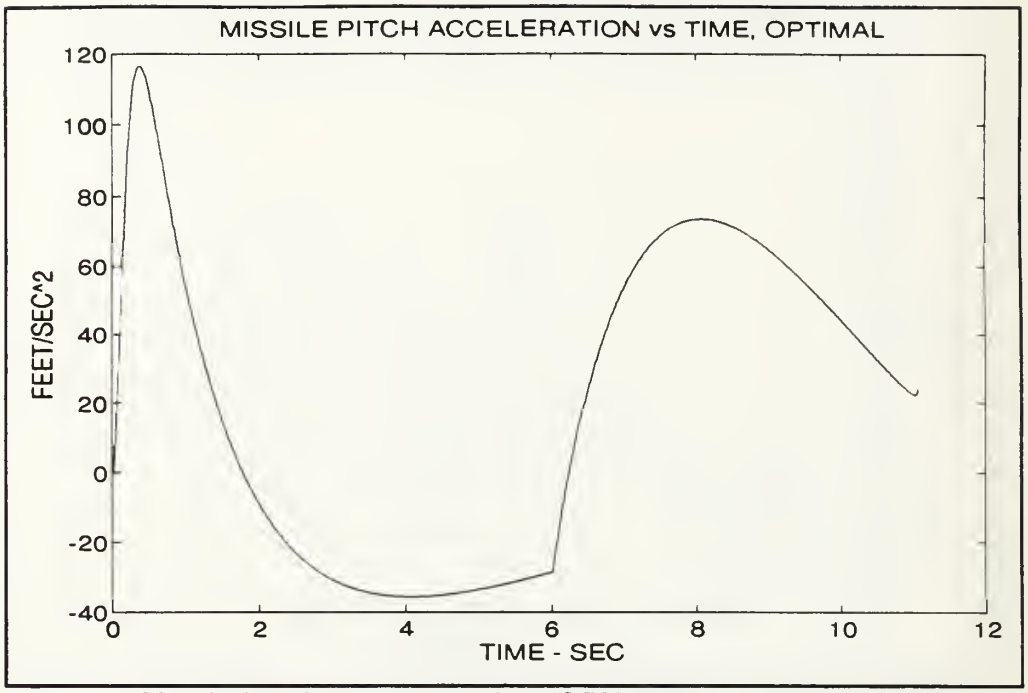

Figure 5.20 Missile Pitch Acceleration (OPTIMAL)

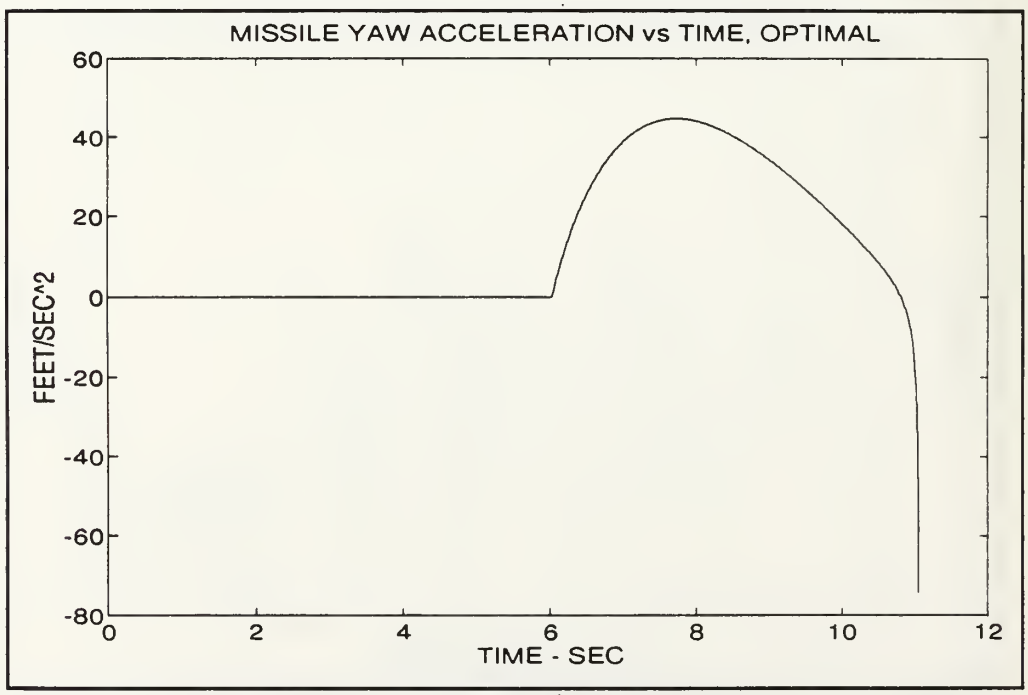

Figure 5.21 Missile Yaw Acceleration (OPTIMAL)

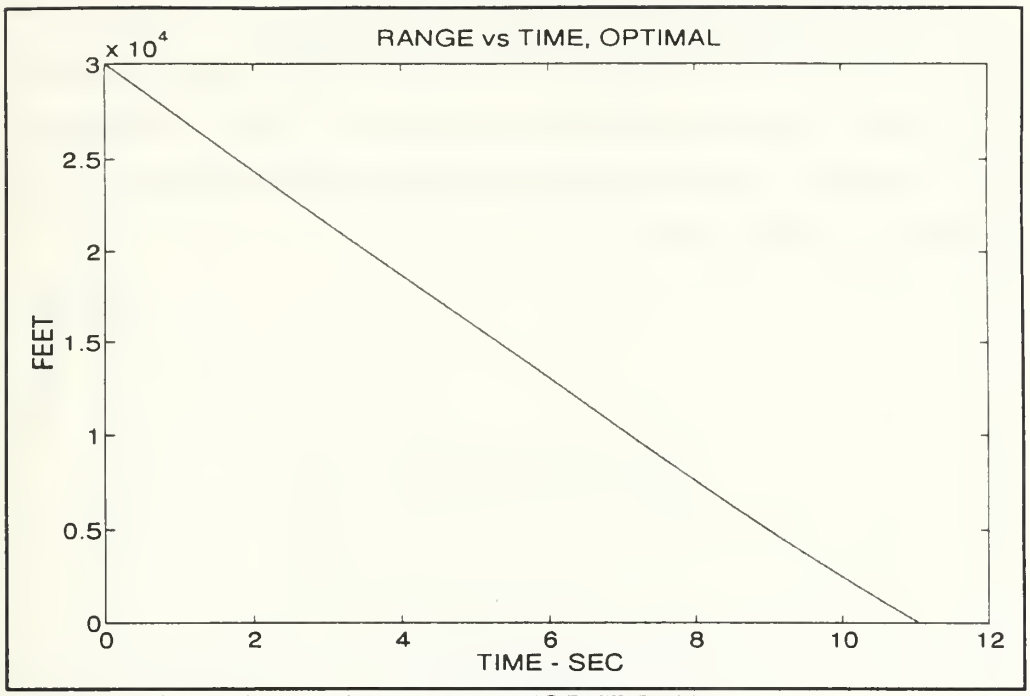

Figure 5.22 Missile To Target Range (OPTIMAL)

2. Scenario #2 (Varying Target Acceleration)

The initial missile/target geometry and the initial conditions are the same as used for the constant acceleration scenario. Namely, the target geometry is shown in Figure 5.1 and the initial conditions are defined by equations 5.4 and 5.5. The target's evasive maneuver may start at any time to go. The target performs the following 3-D maneuver (the accelerations are in feet/sec^2):

, (Eq 5.4)

$$\begin{bmatrix} \ddot{x}_t \\ \ddot{y}_t \\ \ddot{z}_t \end{bmatrix} = \begin{bmatrix} 3 \times g \times \sin(\gamma_{t\_yaw}) \\ 4 \times g \times \cos(\gamma_{t\_yaw}) \\ 3 \times g \times \cos(\gamma_{t\_pitch}) \end{bmatrix}$$

where γ_{t_yaw} and γ_{t_pitch} are the target's yaw and pitch flight path angles, respectively. Figures 5.23 to 5.43 display the results of the three dimensional simulation for this type of evasive maneuver. Figures 5.23 to 5.29 relate to the proportional navigation (PROPNAV) guidance law. Figures 5.30 to 5.36 relate to the augmented proportional (APROPNAV) guidance law. Figures 5.37 to 5.43 relate to the optimal guidance law. Figures 5.25 to 5.29, 5.32 to 5.36 and 5.39 to 5.43 display results, assuming that the target starts its maneuver 6 seconds after the missile entered into the terminal phase of flight.

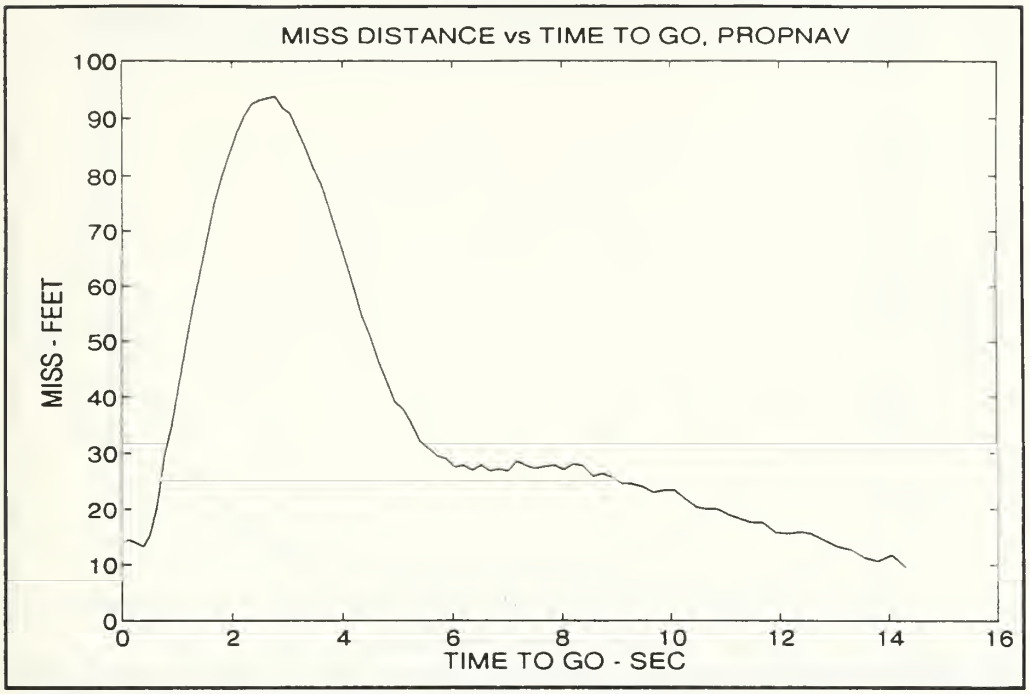

Figure 5.23 Miss Distance vs. Time To Go (PROPNAV)

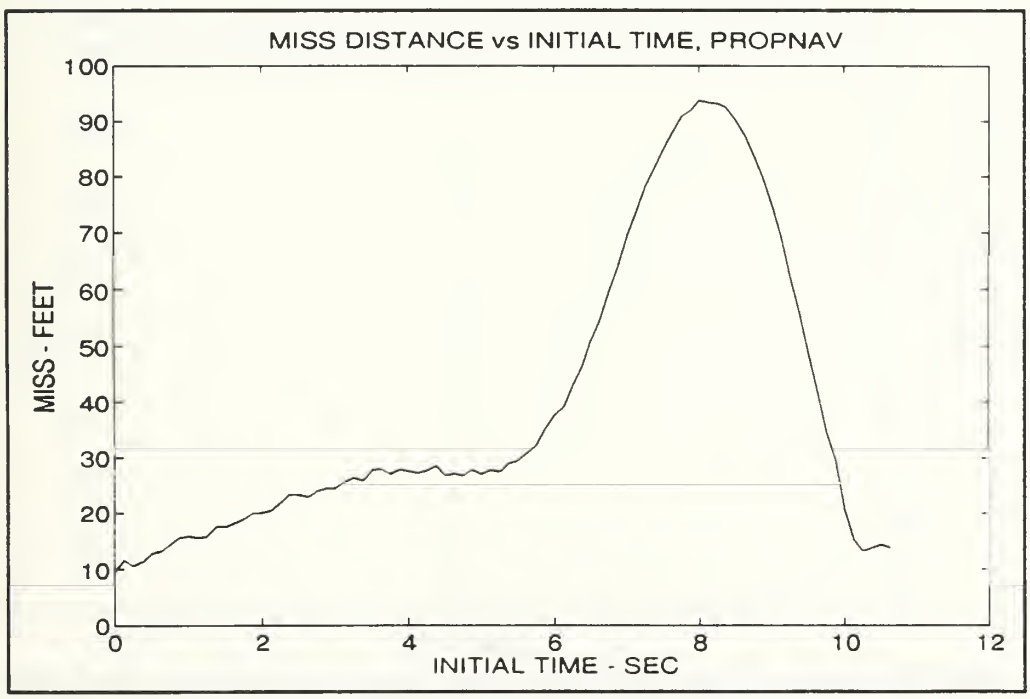

Figure 5.24 Miss Distance vs. Initial Time (PROPNAV)

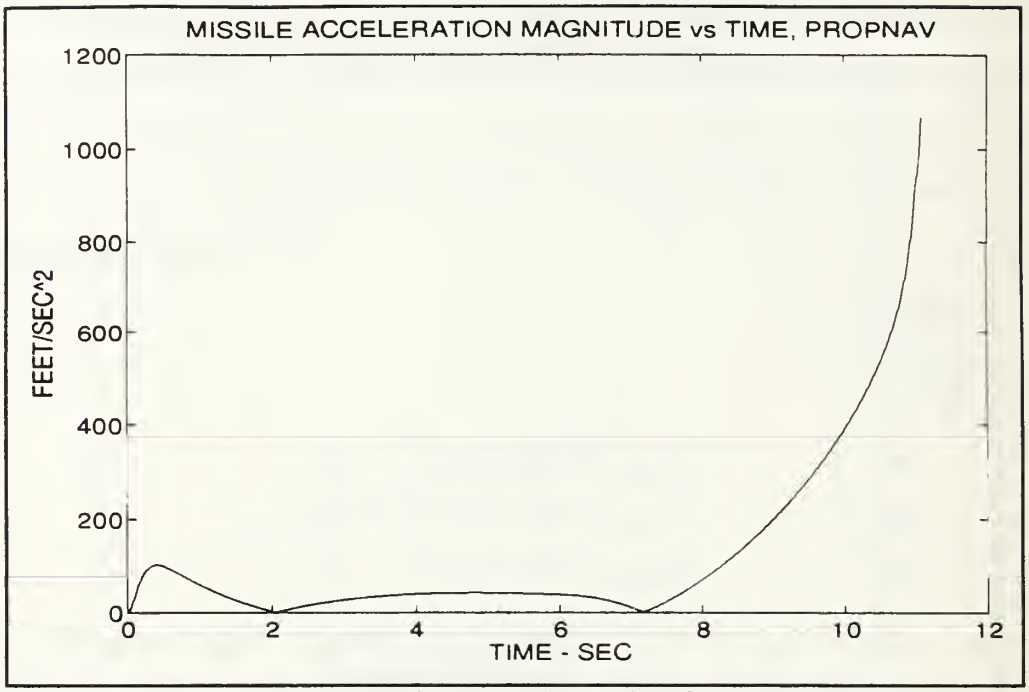

Figure 5.25 Missile Acceleration Magnitude (PROPNAV)

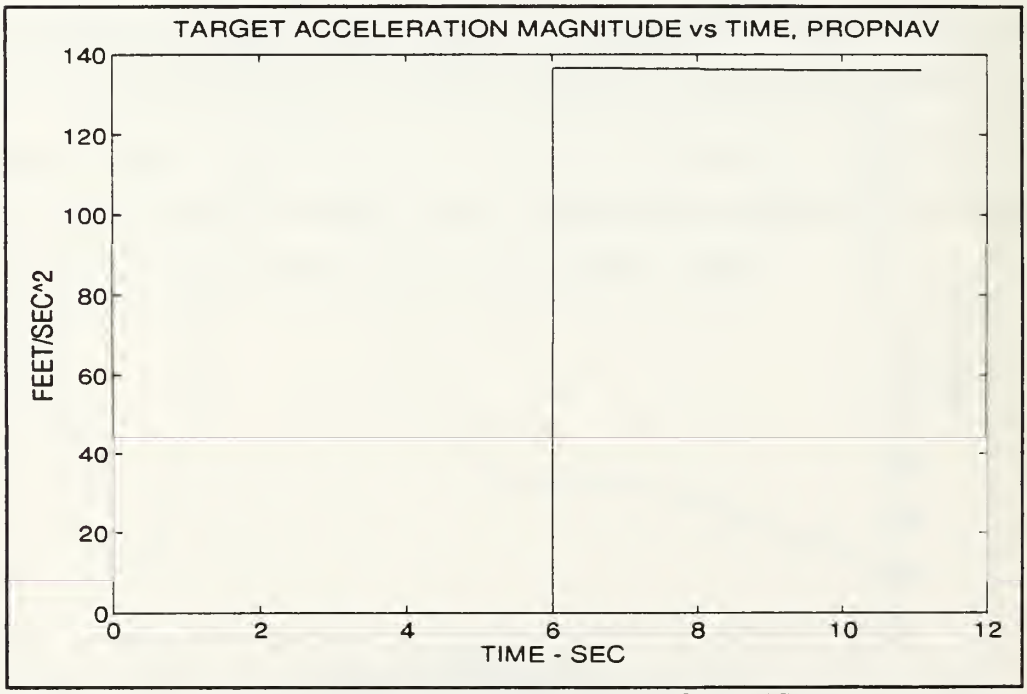

Figure 5.26 Target Acceleration Magnitude (PROPNAV)

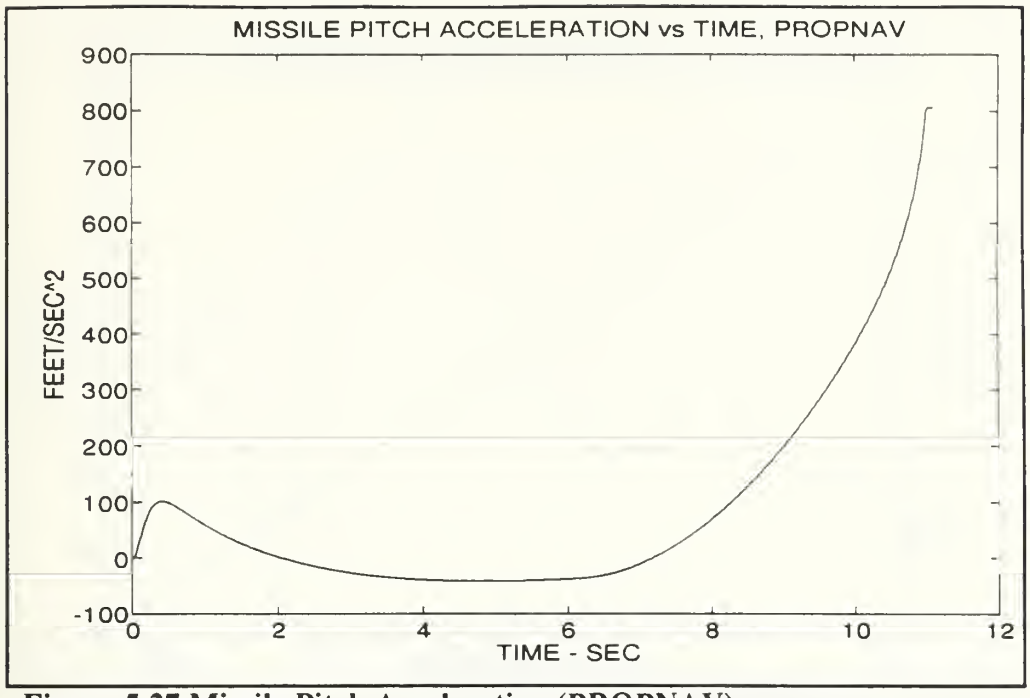

Figure 5.27 Missile Pitch Acceleration (PROPNAV)

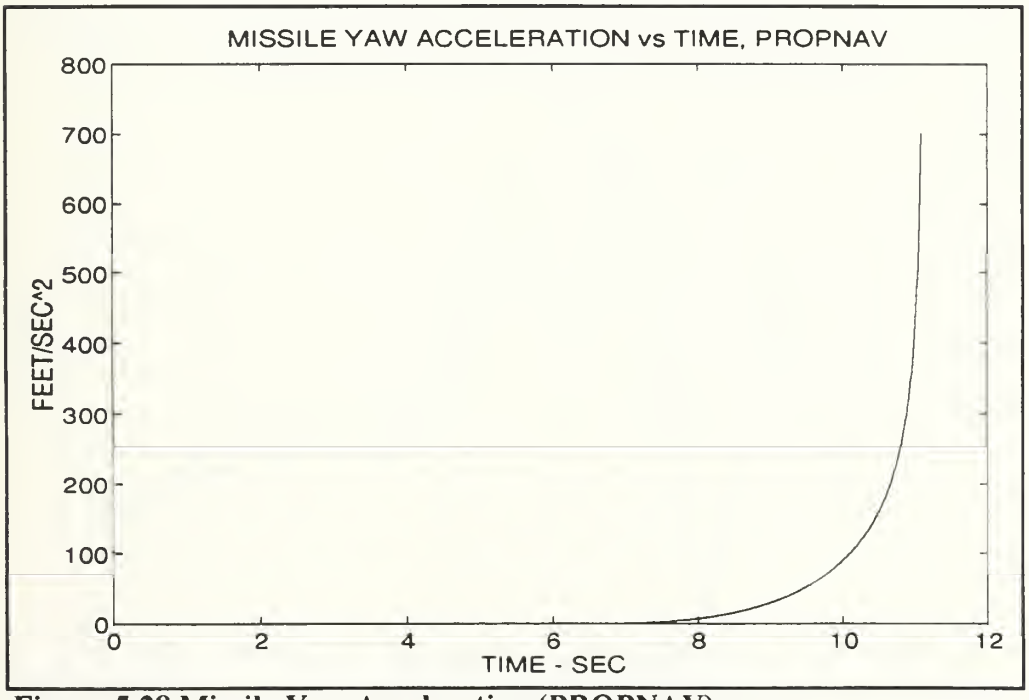

Figure 5.28 Missile Yaw Acceleration (PROPNAV)

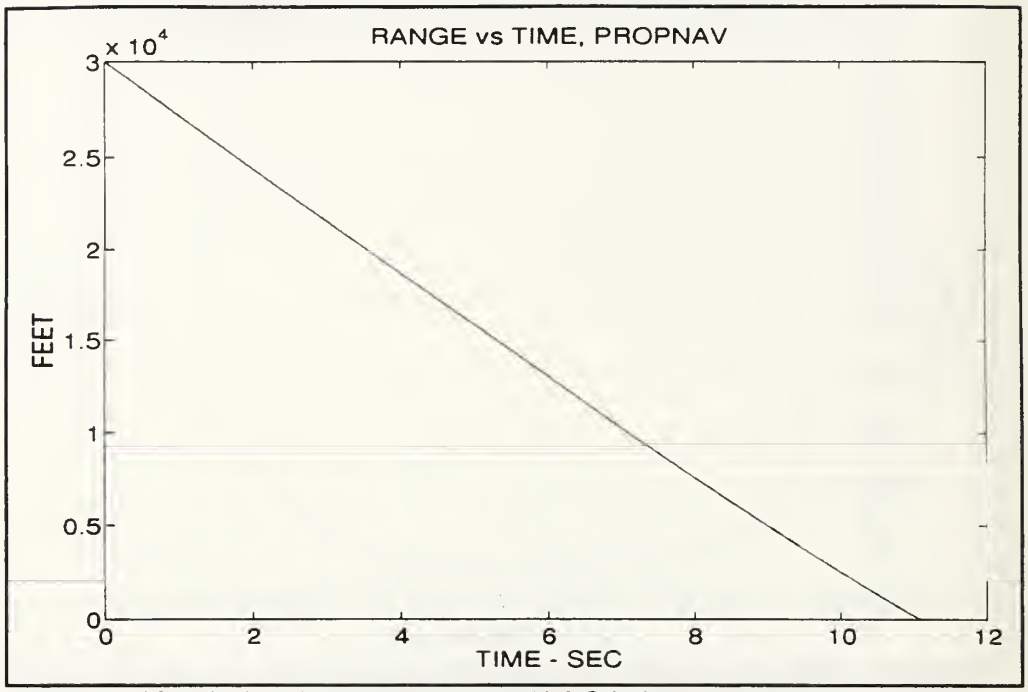

Figure 5.29 Missile To Target Range (PROPNAV)

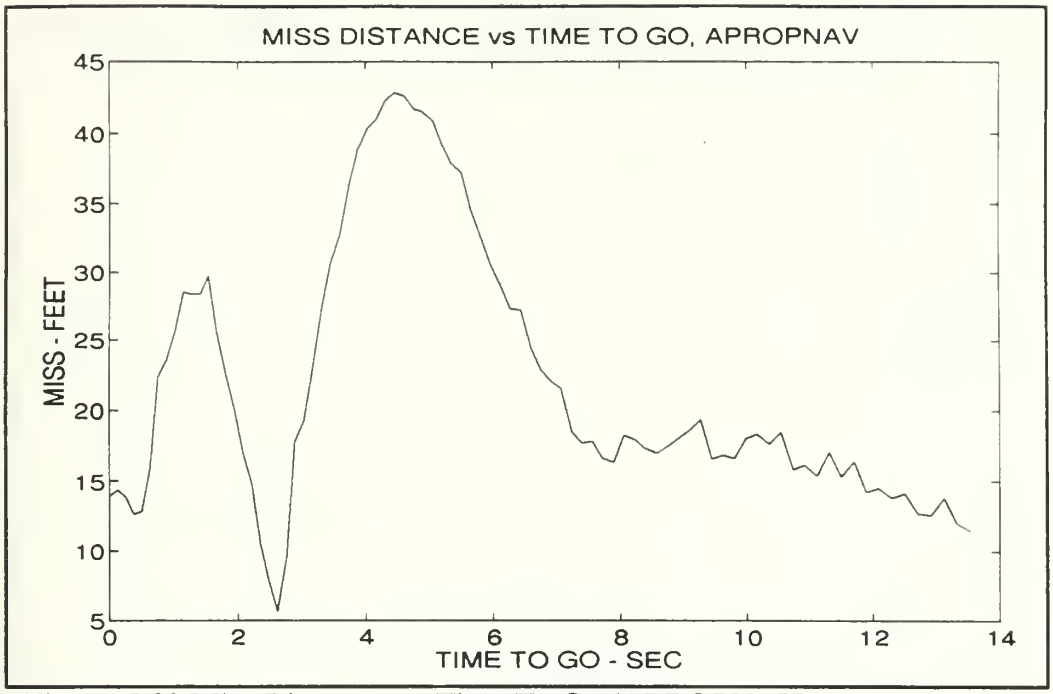

Figure 5.30 Miss Distance vs. Time To Go (APROPNAV)

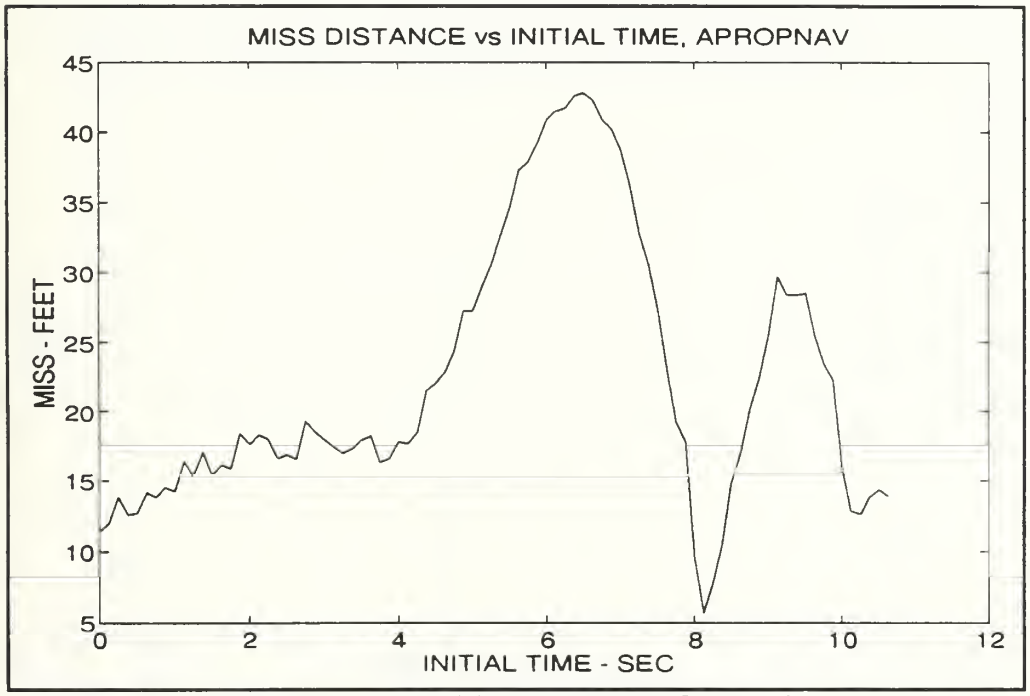

Figure 5.31 Miss Distance vs. Initial Time (APROPNAV)

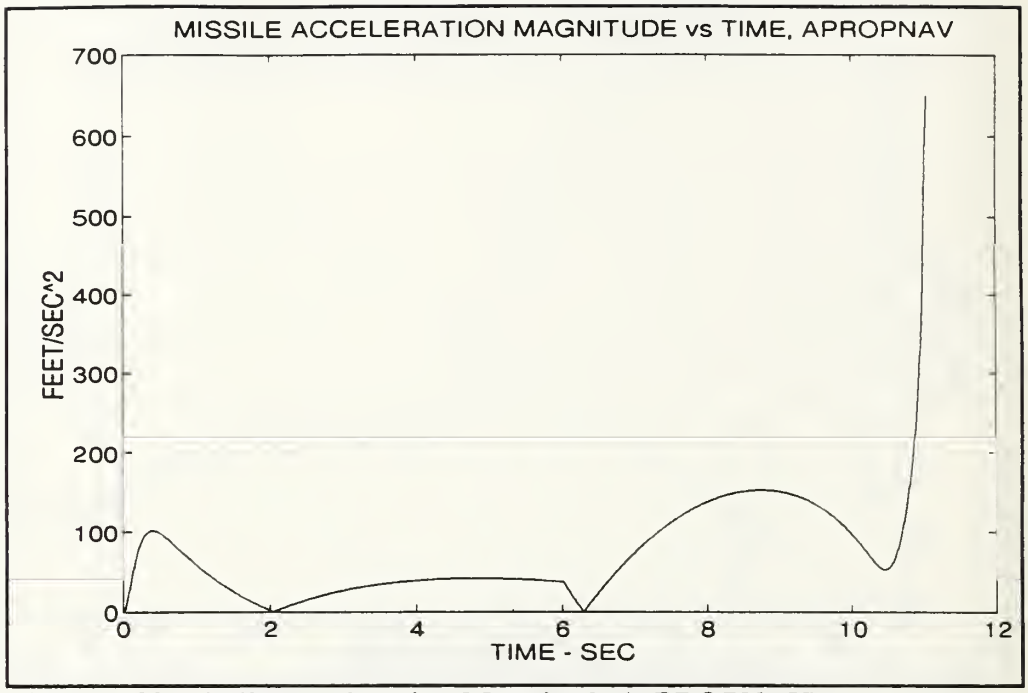

Figure 5.32 Missile Acceleration Magnitude (APROPNAV)

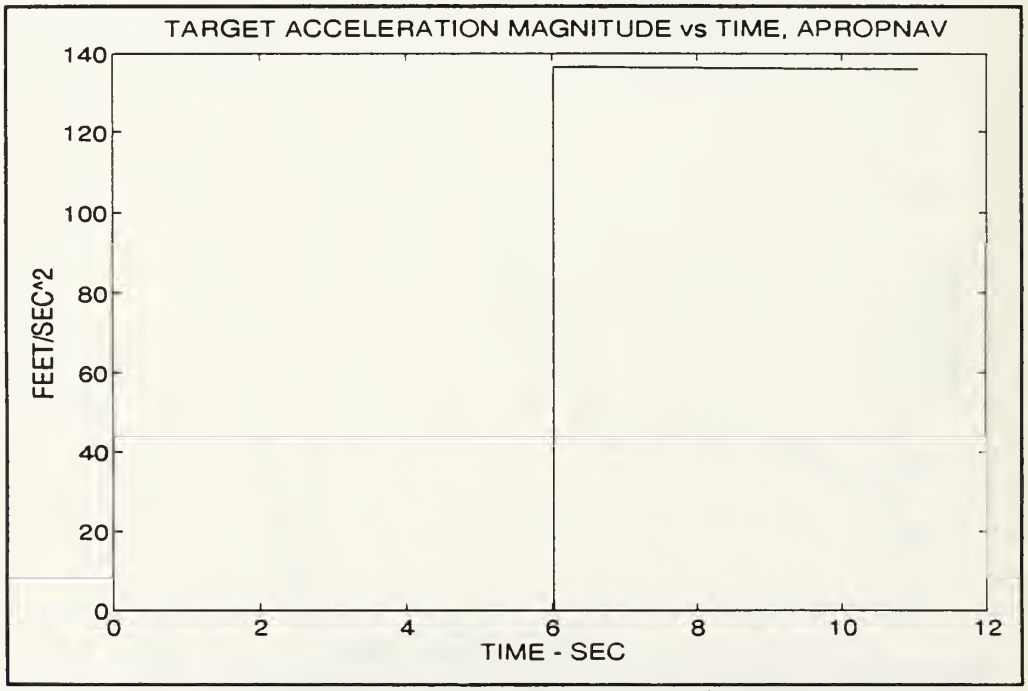

Figure 5.33 Target Acceleration Magnitude (APROPNAV)

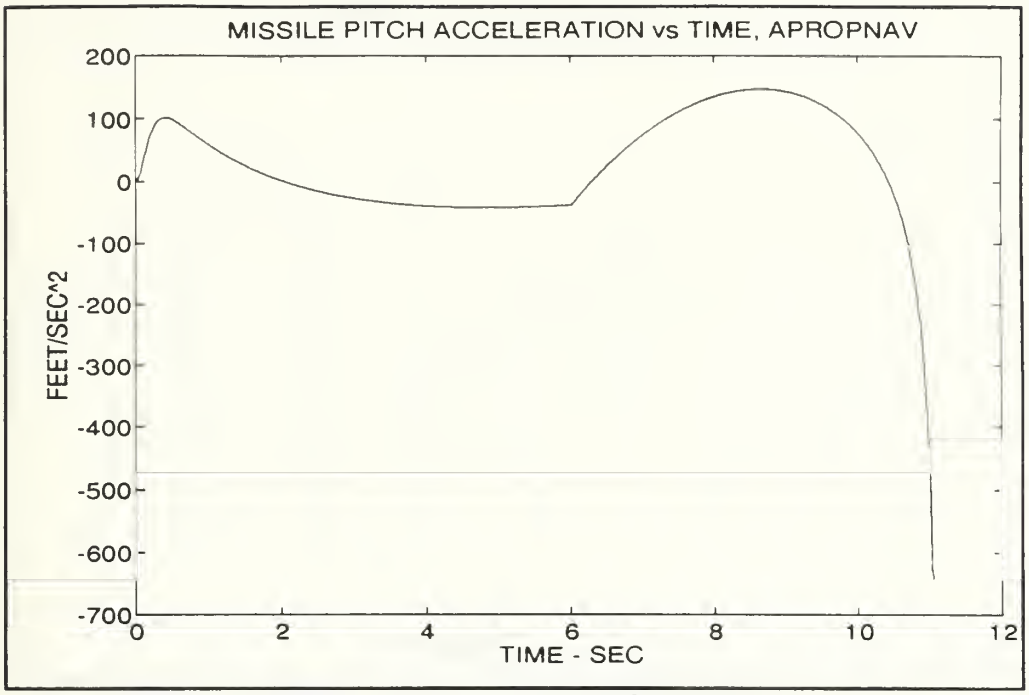

Figure 5.34 Missile Pitch Acceleration (APROPNAV)

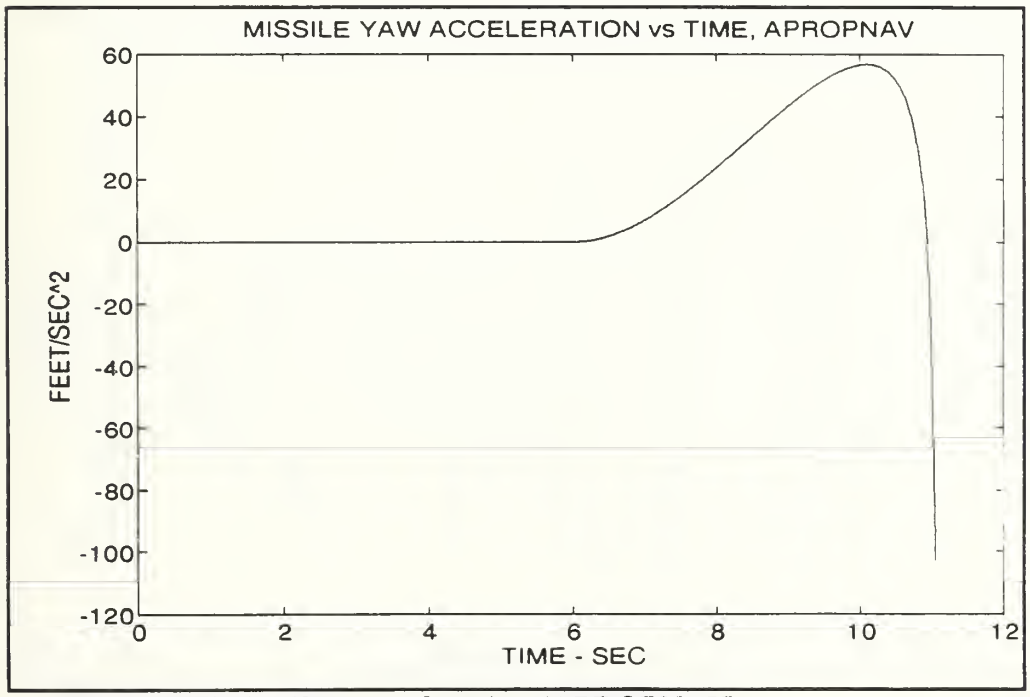

Figure 5.35 Missile Yaw Acceleration (APROPNAV)

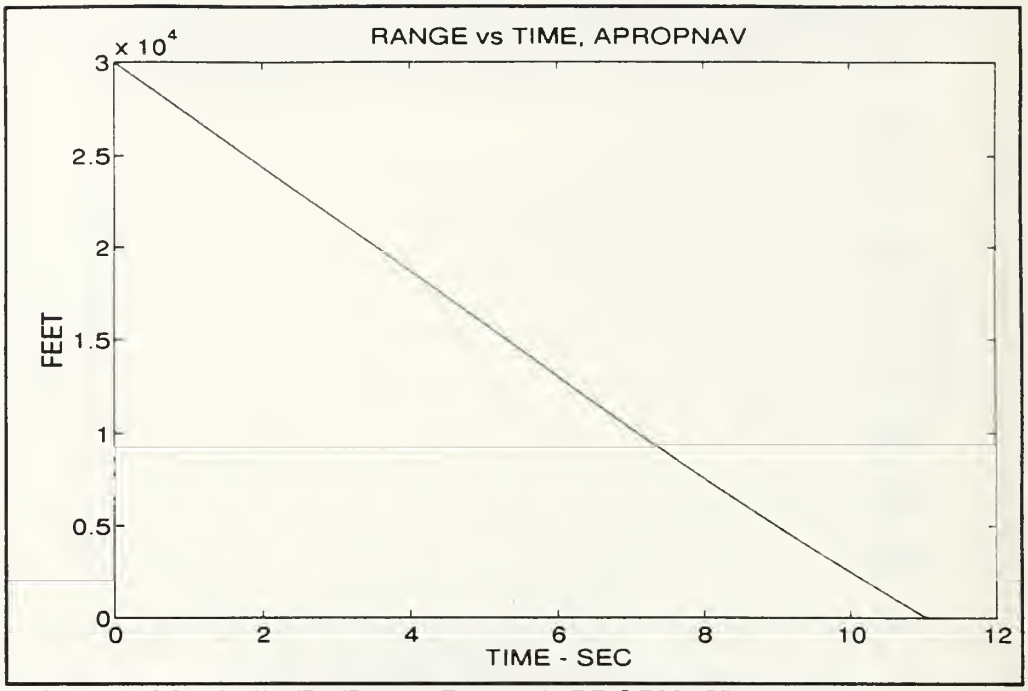

Figure 5.36 Missile To Target Range (APROPNAV)

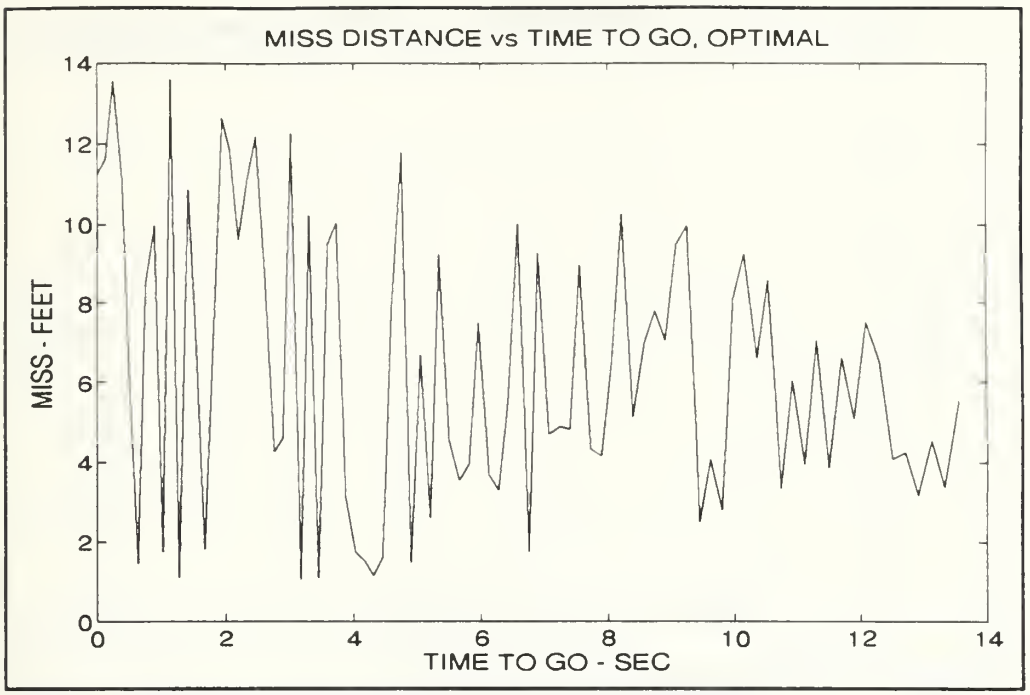

Figure 5.37 Miss Distance vs. Time To Go (OPTIMAL)

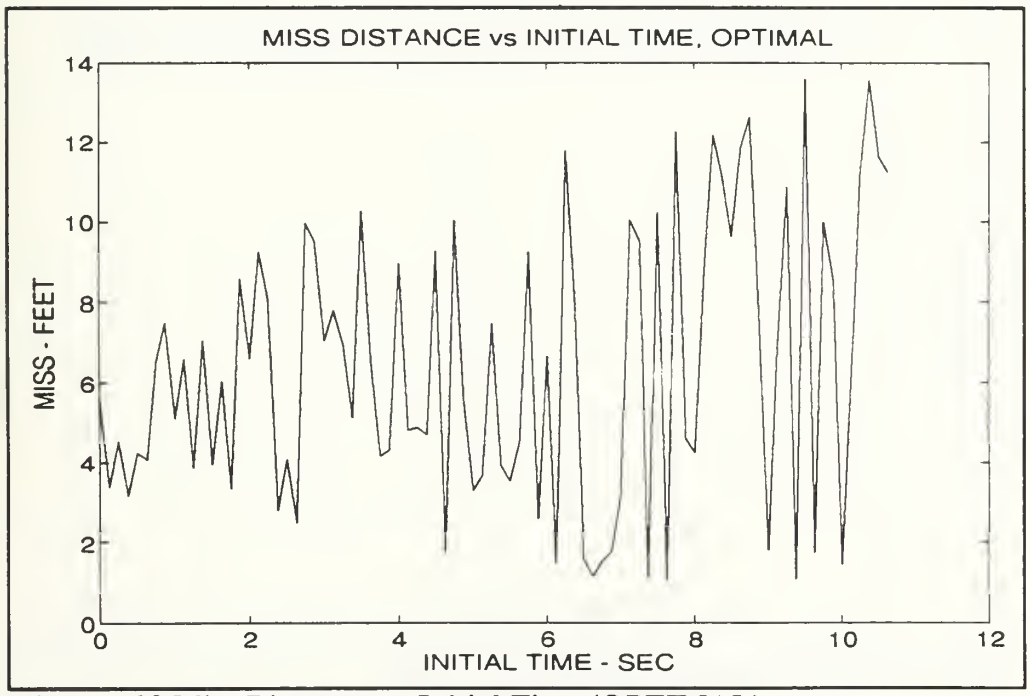

Figure5.38 Miss Distance vs. Initial Time (OPTIMAL)

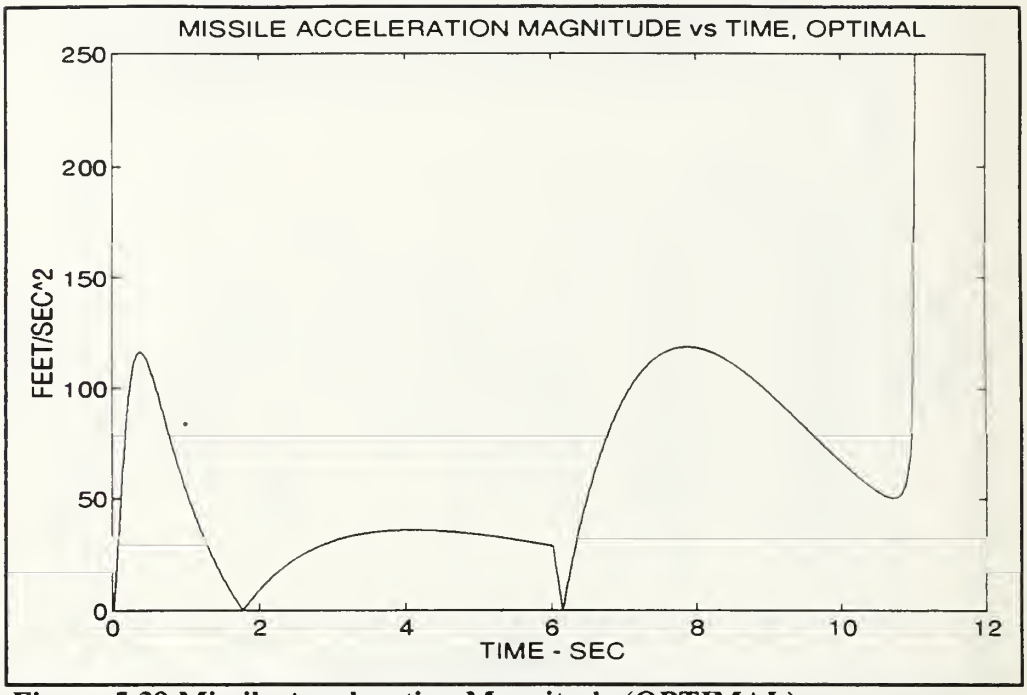

Figure 5.39 Missile Acceleration Magnitude (OPTIMAL)

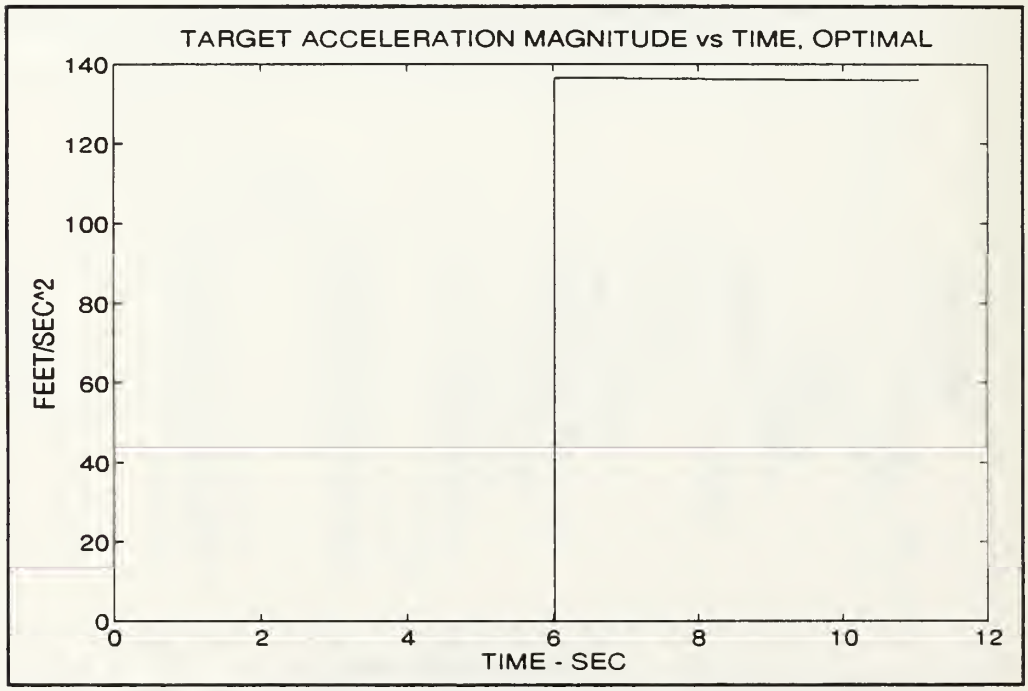

Figure 5.40 Target Acceleration Magnitude (OPTIMAL)

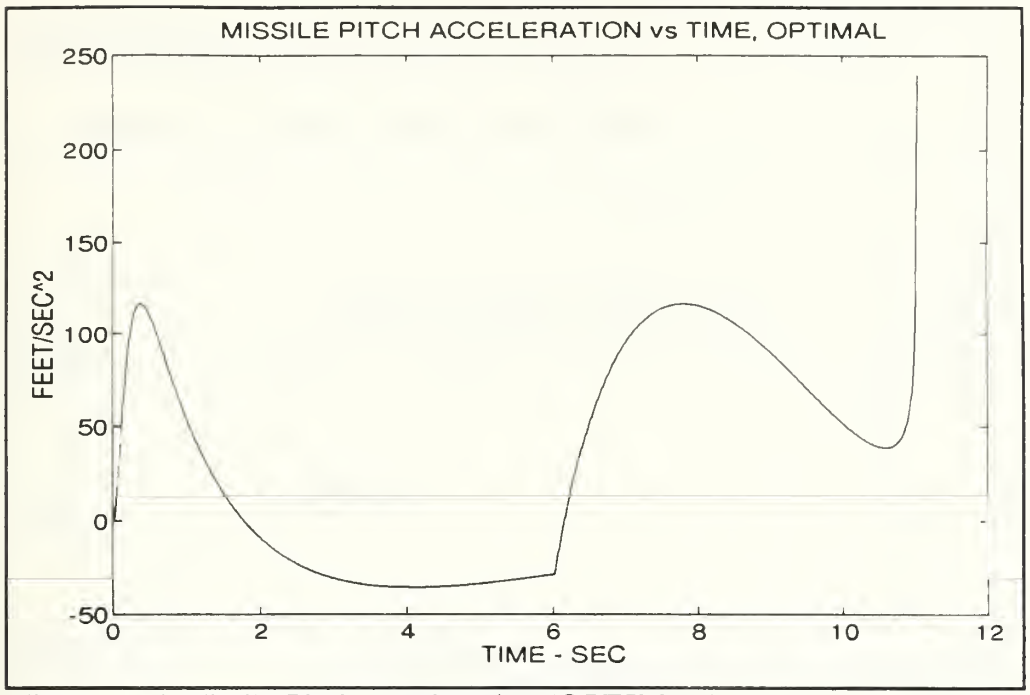

Figure 5.41 Missile Pitch Acceleration (OPTIMAL)

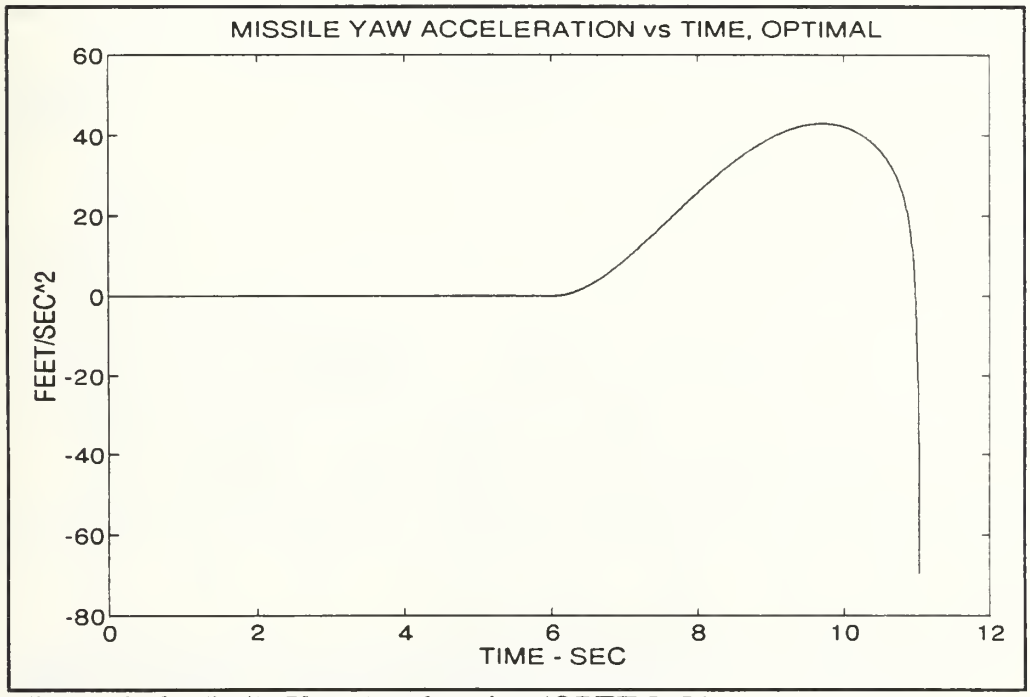

Figure 5.42 Missile Yaw Acceleration (OPTIMAL)

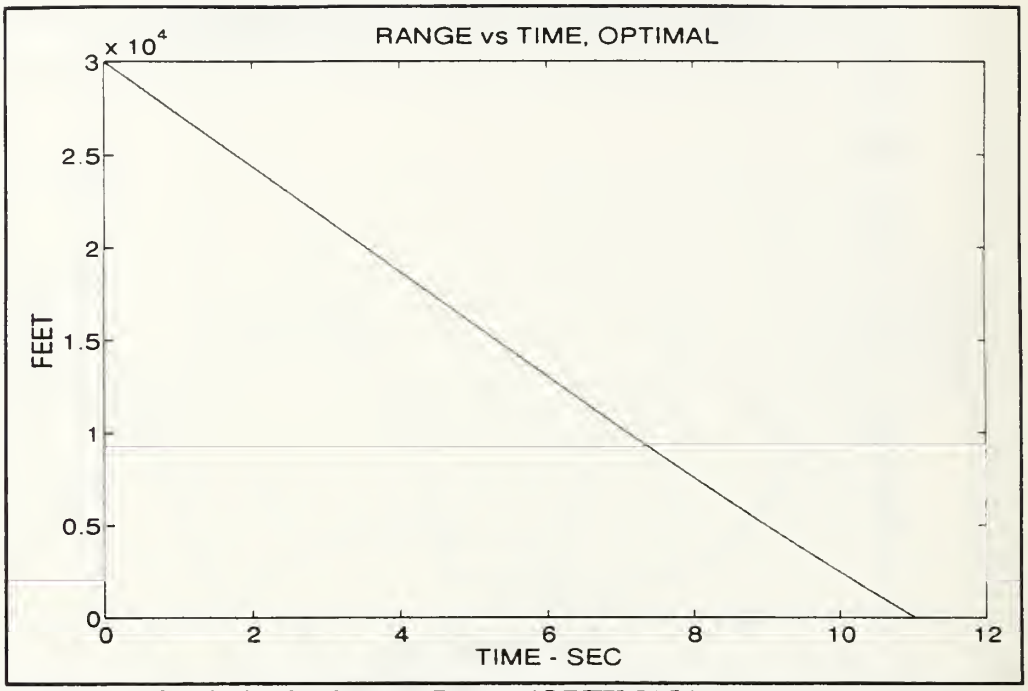

Figure 5.43 Missile To Target Range (OPTIMAL)

VI. CONCLUSIONS AND RECOMENDATIONS

A. CONCLUSIONS

1. Scenario #1 (Constant Target Acceleration)

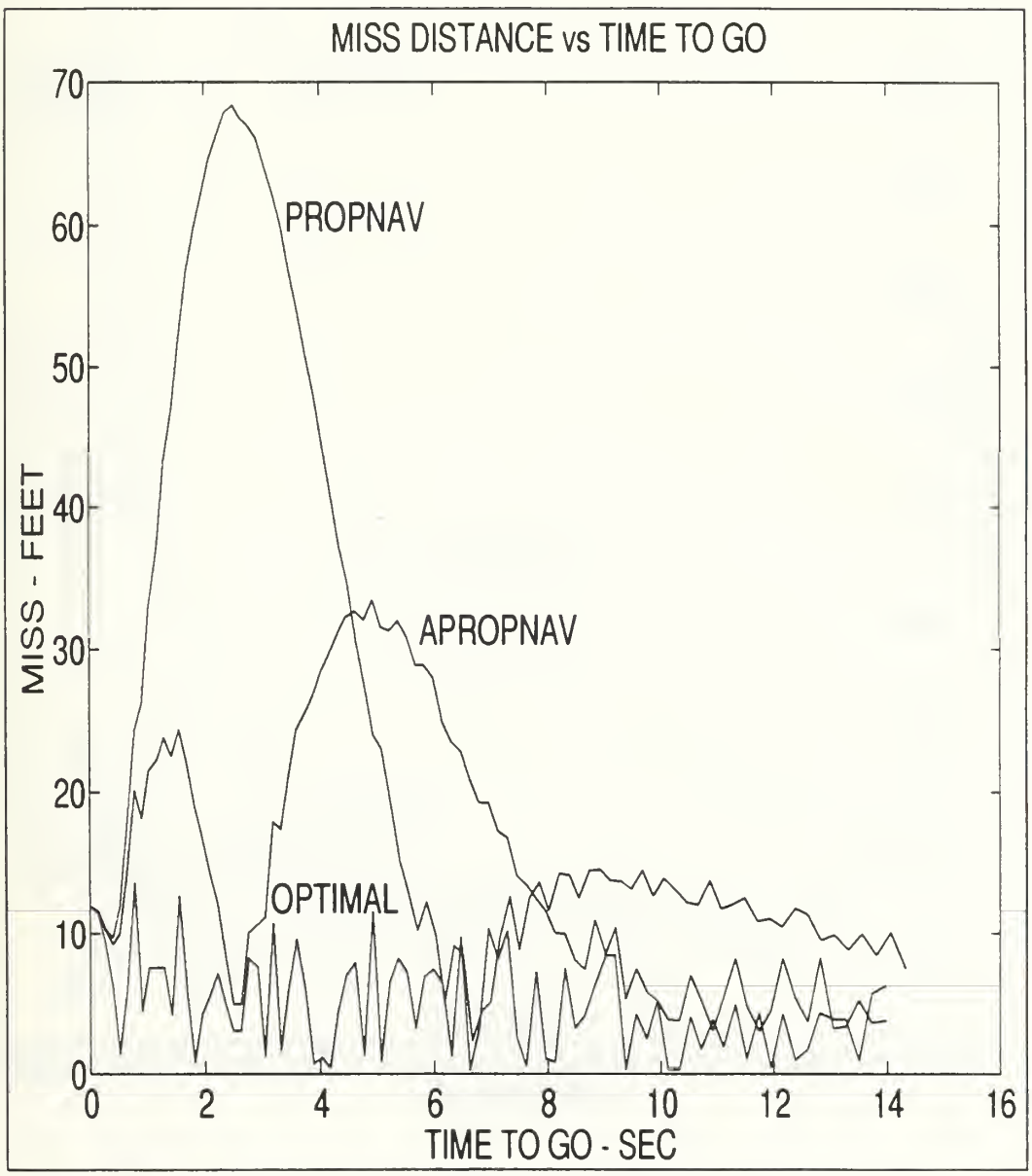

Figure 5.44 Miss Distance Comparison For The Three Guidance Laws (Scenario #1)

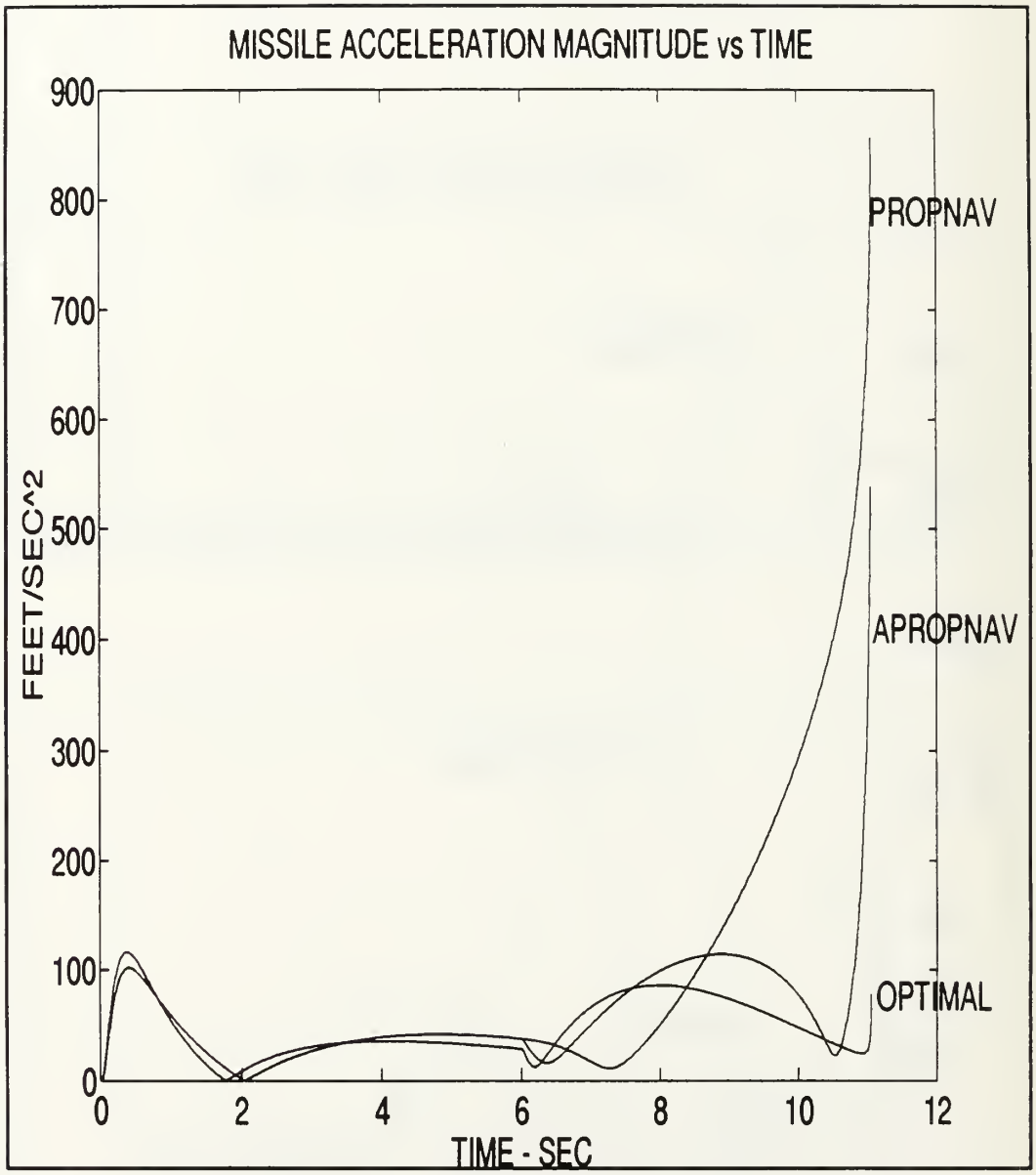

Figure 5.45 Missile Acceleration Comparison For The Three Guidance Laws (Scenario #1)

2. Scenario #2(Varying Target Acceleration)

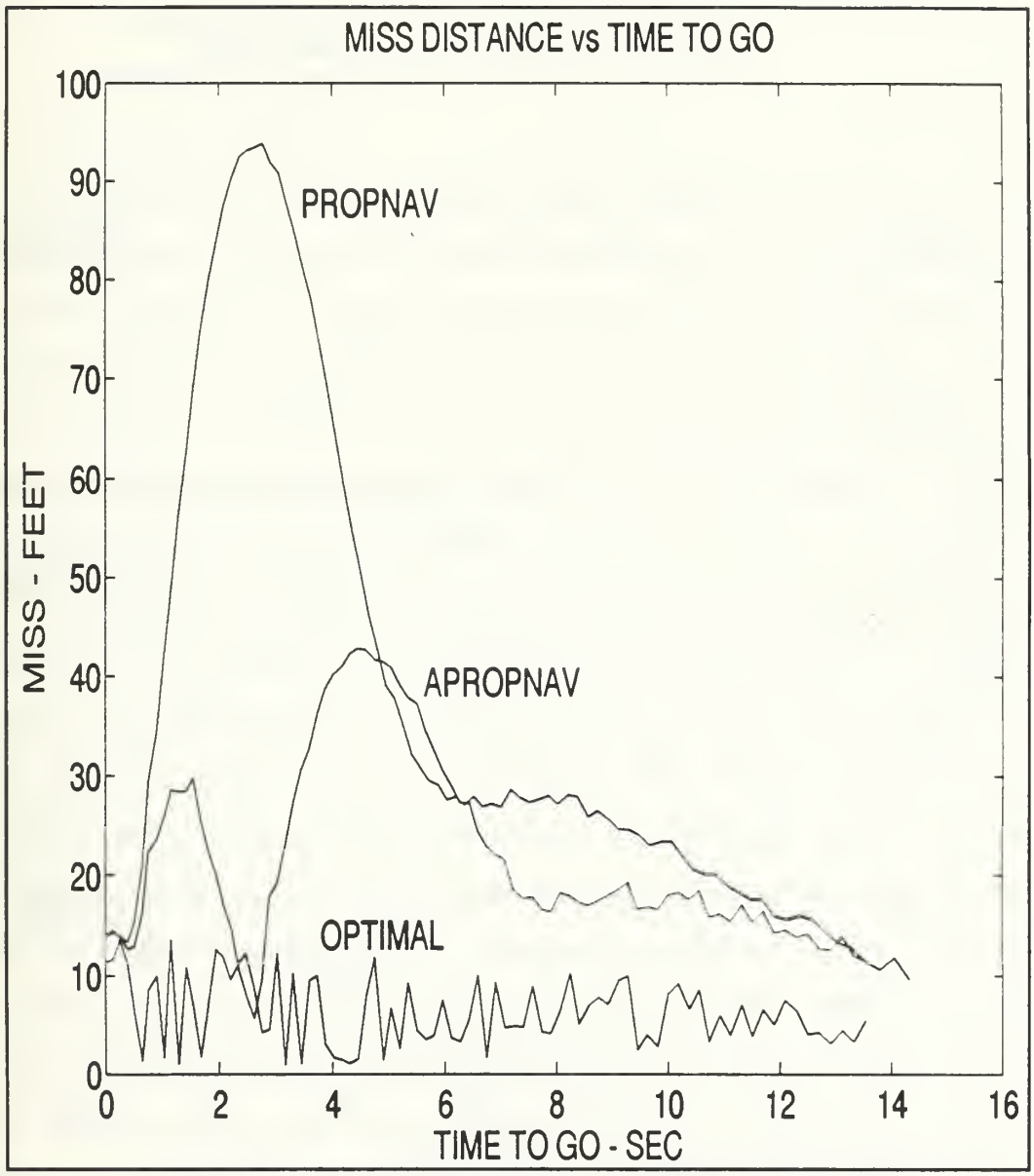

Figure 5.46 Miss Distance Comparison For The Three Guidance Laws (Scenario #2)

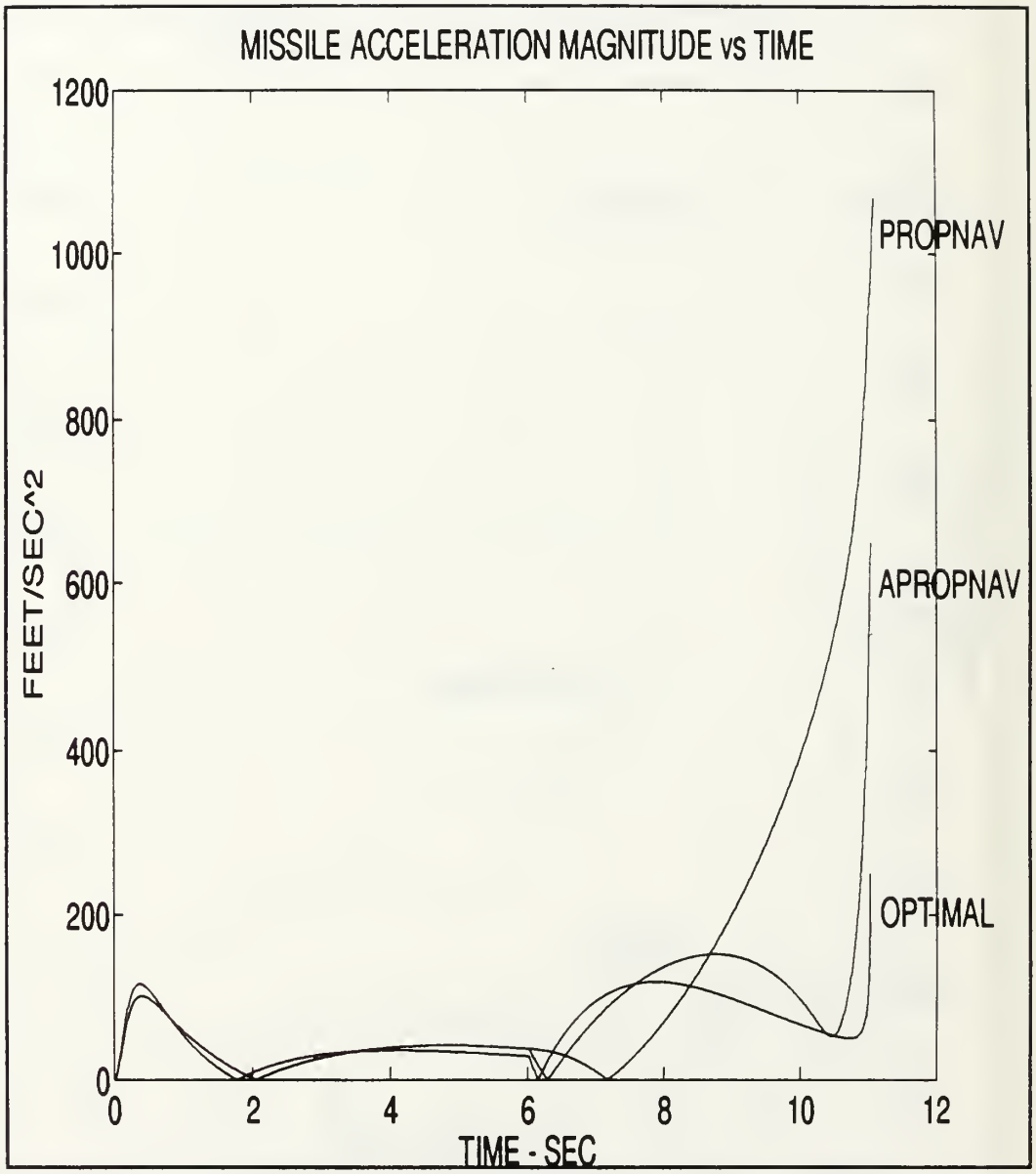

Figure 5.47 Missile Acceleration Comparison For The Three Guidance Laws (Scenario #2)

3. Discussion

The three dimensional miss distance may be improved by estimating the 3-D target's evasive maneuver. One way to estimate the 3-D target acceleration is by utilizing dynamic image processing. Three setups were considered:

1. two cameras,
2. a camera and a radar,
3. only one camera.

In the single camera setup, actual range is achieved by guessing the target's physical dimensions. The target's 3-D motion parameters can be estimated by utilizing two consecutive image frames. The target's acceleration may be computed by taking the second derivative after filtering the target's motion data. Alternatively, the target's 3-D motion may be processed by Kalman filters to estimate its acceleration components. This additional information about the target's behavior is injected in suitable control laws to improve the missile's homing performance.

Proportional navigation, augmented proportional navigation, and optimal guidance laws were derived for use in a three dimensional environment. The classical proportional navigation guidance law tracks a target with good accuracy, especially if the target maneuvers at long time to go. However, when compared with augmented PROPNAV and optimal guidance, PROPNAV requires higher missile acceleration capabilities. A plausible guidance law is one that issues missile's commands proportional to the miss distance that would result if the missile made no further corrections. Augmented proportional navigation was derived using this heuristic argument. For a constant target maneuver, augmented proportional navigation increases the missile percentage of kill. For a non constant evasive maneuver, APROPNAV does not always guarantees less miss distance than PROPNAV. However, APROPNAV requires less missile acceleration capabilities than PROPNAV. Optimal guidance was derived for a missile with a single lag guidance system. Optimal guidance provides compensation for the missile's guidance system dynamics. The optimal guidance law requires the least missile acceleration

capability of the three guidance laws. In fact, this law is derived in order to drive the miss distance to zero while minimizing a performance index made up of the integral of the square of the control. Clearly, the optimal guidance law presents the least miss distance of the three guidance laws. However, it requires a missile with complex signal processing capabilities. The homing capabilities of the missile can be dramatically increased by identifying the target's evasive maneuver and injecting this information into the APROPNAV (especially for a constant target maneuver) or optimal guidance control algorithms. The optimal control algorithm guarantees extraordinary performance. Utilizing optimal guidance, especially against highly responsive targets, can be the difference between failure and success.

B. RECOMENDATIONS

It is recommended to continue this research by simulating the overall system (i.e. estimating the 3-D target's evasive maneuver from two consecutive image frames and injecting this data into the tridimensional missile/target engagement simulation programs developed in this thesis). The simulations developed in this thesis are very generic and easily adapted to different conditions (i.e. for systems with different dynamics and initial conditions). The consequences of the image measurement errors in the target acceleration estimation and ultimately in the miss distance can be investigated. Finally, it is recommended that electronic counter measures (ECCM) be added to the target's evasion maneuver in order to evaluate their effects on the miss distance.

APPENDIX A - MISSILE/TARGET THREE DIMENSIONAL SIMULATION USING PROPORTIONAL NAVIGATION GUIDANCE

```
% Written by: Rui Manuel Alves Francisco
% Date: 14 October 1992
% This Program simulates the terminal phase of a 3-D missile/target
% engagement using classical proportional navigation.

clear
clg
% DEFINE CONSTANTS
dt = .01; % Sampling time
Tf = 100; % maximum simulation time
kmax = Tf/dt+1;
n = 3; % navigation constant
N = [n 0
     0 n];
% DEFINE STATE EQUATIONS
% Missile
% Xm = [xm = missile's x coordinate
%       xmd = missile's speed (x coordinate)
%       ym = missile's y coordinate
%       ymd = missile's speed (y coordinate)
%       zm = missile's z coordinate
%       zmd = missile's speed (z coordinate)]
Am = [0 1 0 0 0 0
      0 0 0 0 0 0
```

```
        0 0 0 1 0 0
        0 0 0 0 0 0
        0 0 0 0 0 1
        0 0 0 0 0 0];
Bm = [0 0 0
      1 0 0
      0 0 0
      0 1 0
      0 0 0
      0 0 1];
% Target
% Xt = [xt = target's x coordinate
%       xtd = target's speed (x coordinate)
%       yt = target's y coordinate
%       ytd = target's speed (y coordinate)
%       zt = target's z coordinate
%       ztd = target's speed (z coordinate)]
At = [0 1 0 0 0 0
      0 0 0 0 0 0
      0 0 0 1 0 0
      0 0 0 0 0 0
      0 0 0 0 0 1
      0 0 0 0 0 0];
Bt = [0 0 0
      1 0 0
      0 0 0
      0 1 0
      0 0 0
```

```
      0 0 1];
% Seeker (Radar)
% Xsk = [beta_pitch = seeker's pitch angle
%        betad_pitch = seeker's pitch angle rate
%        beta_yaw = seeker's yaw angle
%        betad_yaw = seeker's yaw angle rate]
Ask = [ 0    1     0    0
      -100  -20     0    0
         0    0     0    1
         0    0  -100  -20];
Bsk = [ 0 0
       100 0
      0   0
      0 100];
% Guidance System
% Xgs = [a_m_pitch = missile's pitch acceleration
%        a_m_yaw = missile's yaw acceleration]
Ags = [-1 0
        0 -1];
Bgs = [1 0
       0 1];
% INITIALIZE STATE VARIABLES (when the missile enters into the terminal
 phase of flight)
% Missile
Xm(:,1) = [    0                % The missile is in a collision triangle
            2828                % with the target when the missile enters into
               0                % the terminal phase of flight
            1000
```

```
                0
            47.1339];
% Target
Xt(:,1) = [30000
      0
      0
   1000
    500
      0];
% DISCRETE REPRESENTATION
[PHIm,DELm] = c2d(Am,Bm,dt);
[PHIt,DELt] =  c2d(At,Bt,dt);
[PHIsk,DELsk] = c2d(Ask,Bsk,dt);
[PHIgs,DELgs] = c2d(Ags,Bgs,dt);
%LINE OF SIGHT (LOS) INFORMATION. INITIAL CONDITIONS.
% Missile
% LAMBDA_m = Missile's LOS from the global coordinate system
% LAMBDA_m = [LAMBDA_m_pitch = Missile's pitch LOS angle
%                    LAMBDA_m_yaw = Missile's yaw LOS angle]
LAMBDA_m(:,1) = [atan2(Xm(5,1),sqrt(Xm(1,1)^2+Xm(3,1)^2));
                    atan2(Xm(3,1),Xm(1,1))];
% Target
% LAMBDA_t = Target's LOS from the global coordinate system
% LAMBDA_t = [LAMBDA_t_pitch = Target's pitch LOS angle
%                    LAMBDA_t_yaw = Target's yaw LOS angle]
LAMBDA_t(:,1) = [atan2(Xt(5,1),sqrt(Xt(1,1)^2+Xt(3,1)^2));
                    atan2(Xt(3,1),Xt(1,1))];
% LOS from Missile to Target
```

```
% LAMBDA = LOS from Missile to Target.
% LAMBDA = [LAMBDA_pitch = LOS angle in pitch
%                 LAMBDA_yaw = LOS angle in yaw];
LAMBDA(:,1) = [atan2((Xt(5,1)-Xm(5,1)),sqrt((Xt(1,1)-Xm(1,1))^2
                    +(Xt(3,1)-Xm(3,1))^2));
                    atan2((Xt(3,1)-Xm(3,1)),abs(Xt(1,1)-Xm(1,1)))];
% MISSILE and TARGET FLIGHT PATH ANGLES INFORMATION
% Missile
%GAMMA_m = [GAMMA_m_pitch = Missile's flight path angle in pitch
%                 GAMMA_m_yaw = Missile's flight path angle in yaw]
GAMMA_m(:,1) = [atan2(Xm(6,1),sqrt(Xm(2,1)^2+Xm(4,1)^2));
                    atan2(Xm(4,1),Xm(2,1))];
% Target
%GAMMA_t = [GAMMA_t_pitch = Target's flight path angle in pitch
%                 GAMMA_t_YAW = Target's flight path angle in yaw]
GAMMA_t(:,1) = [atan2(Xt(6,1),sqrt(Xt(2,1)^2+Xt(4,1)^2));
                    atan2(Xt(4,1),Xt(2,1))];
% RANGE INFORMATION
% Missile
% Rm = Missile's range
Rm(1) = sqrt(Xm(1,1)^2 + Xm(3,1)^2 + Xm(5,1)^2);
% Target
% Rt = Target's range
Rt(1) = sqrt(Xt(1,1)^2 + Xt(3,1)^2 + Xt(5,1)^2);
% Missile/Target relative distance
% R = [Rmtx = Missile/Target x coordinate range
%      Rmty = Missile/Target y coordinate range
%      Rmtz = Missile/Target z coordinate range
```

```
% Rmt = Missile/Target relative distance(Rmt=sqrt(Rmtx^2+Rmty^2+Rmtz^2))]
R(:,1) = [Xt(1,1)-Xm(1,1)
          Xt(3,1)-Xm(3,1)
          Xt(5,1)-Xm(5,1)
    sqrt((Xt(1,1)-Xm(1,1))^2+(Xt(3,1)-Xm(3,1))^2+(Xt(5,1)-Xm(5,1))^2)];
% VELOCITY INFORMATION
% Missile
% Vm = Missile's Speed
Vm(1) = sqrt(Xm(2,1)^2+Xm(4,1)^2+Xm(6,1)^2);
% Target
% Vt = Target's Speed
Vt(1) = sqrt(Xt(2,1)^2+Xt(4,1)^2+Xt(6,1)^2);
% Speed along the pitch and yaw LOS. Pitch and yaw closing speeds
Vt_pitch(1) = Vt(1)*cos(LAMBDA(2,1)-GAMMA_t(2,1));
Vm_pitch(1) = Vm(1)*cos(LAMBDA(2,1)-GAMMA_m(2,1));
Vc_pitch(1) = Vm_pitch(1)*cos(GAMMA_m(1,1)-LAMBDA(1,1))
                -Vt_pitch(1)*cos(GAMMA_t(1,1)-LAMBDA(1,1));
Vt_yaw(1) = Vt(1)*cos(GAMMA_t(1,1));
Vm_yaw(1) = Vm(1)*cos(GAMMA_m(1,1));
Vc_yaw(1) = Vm_yaw(1)*cos(GAMMA_m(2,1)-LAMBDA(2,1))
                -Vt_yaw(1)*cos(GAMMA_t(2,1)-LAMBDA(2,1));
Vc = [Vc_pitch(1) 0
      0           Vc_yaw(1)];
% SEEKER and GUIDANCE SYSTEM INITIAL CONDITIONS and INPUTS.
% Seeker
Xsk(:,1) = [0
            0
            0
```

```
                0];
Usk(:,1) = LAMBDA(:,1); % Seeker input
% Guidance System
Xgs(:,1) = [0
                0];
Ugs(:,1) = N*Vc*[Xsk(2,1) % Guidance system input
                        Xsk(4,1)];
% TIME
% Time = Time vector
TIME(1) = 0;
% Tgo = Time to go
Tgo(1) = R(4,1)/Vc_pitch(1);
% SIMULATE THE SYSTEM
for ti = 0:.25:21.25
  for i = 1:kmax-1
% Calculate components of the missile's pitch acceleration
a_mx_pitch(i) = -(Xgs(1,i)*sin(LAMBDA(1,i))*cos(LAMBDA(2,i)));
a_my_pitch(i) = -(Xgs(1,i)*sin(LAMBDA(1,i))*sin(LAMBDA(2,i)));
a_mz_pitch(i) = Xgs(1,i)*cos(LAMBDA(1,i));
% Calculate components of the missile's yaw acceleration
a_mx_yaw(i) = -Xgs(2,i)*sin(LAMBDA(2,i));
a_my_yaw(i) = Xgs(2,i)*cos(LAMBDA(2,i));
% Compute overall missile acceleration
a_mx(i) = a_mx_pitch(i)+a_mx_yaw(i);
a_my(i) = a_my_pitch(i)+a_my_yaw(i);
a_mz(i) = a_mz_pitch(i);
am(:,i) = [a_mx(i)
           a_my(i)
```

```
            a_mz(i)];
% Compute missile's acceleration magnitude
a_m(i) = sqrt(a_mx(i)^2 + a_my(i)^2 + a_mz(i)^2);
% Generate target's evasive maneuver (we assume that these accelerations, along
% the three cartesian axis, are estimated using the missile's image processing
% capabilities)
if TIME(i) >= ti/2 % target starts evasive maneuver
  a_tx(i) = 3*32.2*sin(GAMMA_t(2,i));
  a_ty(i) = 4*32.2*cos(GAMMA_t(2,i));
  a_tz(i) = 3*32.2*cos(GAMMA_t(1,i));
else
  a_tx(i) = 0.0;
  a_ty(i) = 0.0;
  a_tz(i) = 0.0;
end
at(:,i) = [a_tx(i)
           a_ty(i)
           a_tz(i)];
% Compute magnitude of the target's acceleration
a_t(i) = sqrt(a_tx(i)^2 + a_ty(i)^2 + a_tz(i)^2);
% Update missile states
Xm(:,i+1) = PHIm*Xm(:,i)+DELm*am(:,i);
% Update target states
Xt(:,i+1) = PHIt*Xt(:,i)+DELt*at(:,i);
% Update seeker states
Xsk(:,i+1) = PHIsk*Xsk(:,i)+DELsk*Usk(:,i);
% Update Guidance System states
Xgs(:,i+1) = PHIgs*Xgs(:,i)+DELgs*Ugs(:,i);
```

```
% Limit yaw and pitch accelerations to 25 g's
 if abs(Xgs(1,i+1)) > 805.0
   Xgs(1,i+1) = 805.0 *sign(Xgs(1,i+1));
 end
 if abs(Xgs(2,i+1)) > 805.0
  Xgs(2,i+1) = 805.0 *sign(Xgs(2,i+1));
 end
% Update LOS angles
LAMBDA(:,i+1) = [atan2((Xt(5,i+1)-Xm(5,i+1)),sqrt((Xm(1,i+1)-Xt(1,i+1))^2
                    +(Xm(3,i+1)-Xt(3,i+1))^2));
                atan2((Xt(3,i+1)-Xm(3,i+1)),(abs(Xm(1,i+1)-Xt(1,i+1))))];
Usk(:,i+1) = LAMBDA(:,i+1);
LAMBDA_m(:,i+1) = [atan2(Xm(5,i+1),sqrt(Xm(1,i+1)^2+Xm(3,i+1)^2));
                    atan2(Xm(3,i+1),Xm(1,i+1))];
LAMBDA_t(:,i+1) = [atan2(Xt(5,i+1),sqrt(Xt(1,i+1)^2+Xt(3,i+1)^2));
                    atan2(Xt(3,i+1),Xt(1,i+1))];
% Update flight path angles
GAMMA_m(:,i+1) = [atan2(Xm(6,i+1),sqrt(Xm(2,i+1)^2+Xm(4,i+1)^2));
                    atan2(Xm(4,i+1),Xm(2,i+1))];
GAMMA_t(:,i+1) = [atan2(Xt(6,i+1),sqrt(Xt(2,i+1)^2+Xt(4,i+1)^2));
                    atan2(Xt(4,i+1),Xt(2,i+1))];
% Update Range Information
Rm(i+1) = sqrt(Xm(1,i+1)^2 + Xm(3,i+1)^2 + Xm(5,i+1)^2);
Rt(i+1) = sqrt(Xt(1,i+1)^2 + Xt(3,i+1)^2 + Xt(5,i+1)^2);
R(:,i+1) = [Xt(1,i+1)-Xm(1,i+1);
            Xt(3,i+1)-Xm(3,i+1);
            Xt(5,i+1)-Xm(5,i+1);
            sqrt((Xt(1,i+1)-Xm(1,i+1))^2+(Xt(3,i+1)-Xm(3,i+1))^2
```

```
            +(Xt(5,i+1)-Xm(5,i+1))^2)];
% Update Velocity Information
Vm(i+1) = sqrt(Xm(2,i+1)^2+Xm(4,i+1)^2+Xm(6,i+1)^2);
Vt(i+1) = sqrt(Xt(2,i+1)^2+Xt(4,i+1)^2+Xt(6,i+1)^2);
Vt_pitch(i+1) = Vt(i+1)*cos(LAMBDA(2,i+1)-GAMMA_t(2,i+1));
Vm_pitch(i+1) = Vm(i+1)*cos(LAMBDA(2,i+1)-GAMMA_m(2,i+1));
Vc_pitch(i+1) = Vm_pitch(i+1)*cos(GAMMA_m(1,i+1)-LAMBDA(1,i+1))
                -Vt_pitch(i+1)*cos(GAMMA_t(1,i+1)-LAMBDA(1,i+1));
Vt_yaw(i+1) = Vt(i+1)*cos(GAMMA_t(1,i+1));
Vm_yaw(i+1) = Vm(i+1)*cos(GAMMA_m(1,i+1));
Vc_yaw(i+1) = Vm_yaw(i+1)*cos(GAMMA_m(2,i+1)-LAMBDA(2,i+1))
                -Vt_yaw(i+1)*cos(GAMMA_t(2,i+1)-LAMBDA(2,i+1));
Vc = [Vc_pitch(i+1) 0
      0             Vc_yaw(i+1)];
% Update guidance system input
Ugs(:,i+1) = N*Vc*[Xsk(2,i+1)
                   Xsk(4,i+1)];
% Update Time/time to go
TIME(i+1) = TIME(i)+dt;
Tgo(i+1) = R(4,i+1)/Vc_pitch(i+1);
% Check for closest point
if (R(4,i) < R(4,i+1)),break,end
end;
% Save information for plotting and evaluation
R1(4*ti+1) = R(4,i); % miss distance
Ti(4*ti+1) = ti/2;% starting time of evasive maneuver (EM)
tgo(4*ti+1) = i*dt-ti/2; % time to go until end of flight
 if ti == 12.0          % Record information for a target that
```

```
                % initialized the evasive maneuver 6 sec after the missile
                % entered into the terminal phase of flight.
  TGO = tgo(49);
  Xseeker = Xsk(:,1:i);
  Xgsys = Xgs(:,1:i);
  lambda_m = LAMBDA_m(:,1:i);
  lambda_t = LAMBDA_t(:,1:i);
  lambda = LAMBDA(:,1:i);
  gamma_m = GAMMA_m(:,1:i);
  gamma_t = GAMMA_t(:,1:i);
  r = R(:,1:i);
  vm = Vm(1:i);
  vt = Vt(1:i);
  vm_pitch = Vm_pitch(1:i);
  vt_pitch = Vt_pitch(1:i);
  vm_yaw = Vm_yaw(1:i);
  vt_yaw = Vt_yaw(1:i);
  vc_pitch = Vc_pitch(1:i);
  vc_yaw = Vc_yaw(1:i);
  tGO = Tgo(1:i);
  a_M = am(:,1:i);
  a_T = at(:,1:i);
  A_t = a_t(1:i);
  A_m = a_m(1:i);
  time = TIME(1:i);
 end
clear R;
R(:,1) = [Xt(1,1)-Xm(1,1);
```

```
            Xt(3,1)-Xm(3,1);
            Xt(5,1)-Xm(5,1);
    sqrt((Xt(1,1)-Xm(1,1))^2+(Xt(3,1)-Xm(3,1))^2+(Xt(5,1)-Xm(5,1))^2)];
end;
save thesis1p343 R1 tgo Ti tGO missile target TGO Xseeker Xgsys lambda_m
 lambda_t lambda gamma_m gamma_t r vm vt vm_pitch vm_yaw vt_pitch
vt_yaw vc_pitch vc_yaw a_M a_T time A_t A_m
PLOTS
% Miss distance information
plot(Ti,R1),title('MISS DISTANCE vs INITIAL TIME, PROPNAV')
xlabel('INITIAL TIME - SEC'),ylabel('MISS - FEET')
print -dps R1ap1
!pstoepsi R1ap1.ps R1ap1.epsi
pause,clg
plot(tgo,R1),title('MISS DISTANCE vs TIME TO GO, PROPNAV')
xlabel('TIME TO GO - SEC'),ylabel('MISS - FEET')
print -dps R1bp1
!pstoepsi R1bp1.ps R1bp1.epsi
pause,clg
% Missile acceleration information
plot(time,A_m),title('MISSILE ACCELERATION MAGNITUDE vs TIME,
                        PROPNAV')
xlabel('TIME - SEC'),ylabel('FEET/SEC^2')
print -dps A_mp1
!pstoepsi A_mp1.ps A_mp1.epsi
pause,clg
plot(time,Xgsys(1,:)),title('MISSILE PITCH ACCELERATION vs TIME,
                        PROPNAV')
```

```
xlabel('TIME - SEC'),ylabel('FEET/SEC^2')
print -dps Xgsys1p1
!pstoepsi Xgsys1p1.ps Xgsys1p1.epsi
pause,clg
plot(time,Xgsys(2,:)),title('MISSILE YAW ACCELERATION vs TIME,
                                PROPNAV')
xlabel('TIME - SEC'),ylabel('FEET/SEC^2')
print -dps Xgsys2p1
!pstoepsi Xgsys2p1.ps Xgsys2p1.epsi
pause,clg
% Target acceleration information
plot(time,A_t),title('TARGET ACCELERATION MAGNITUDE vs TIME,
                        PROPNAV')
xlabel('TIME - SEC'),ylabel('FEET/SEC^2')
print -dps A_tp1
!pstoepsi A_tp1.ps A_tp1.epsi
pause,clg
% Seeker pitch and yaw angles
plot(time,Xseeker(1,:)),title('SEEKER PITCH ANGLE vs TIME, PROPNAV')
xlabel('TIME - SEC'),ylabel('RAD')
print -dps Xseeker1p1
!pstoepsi Xseeker1p1.ps Xseeker1p1.epsi
pause,clg
plot(time,Xseeker(3,:)),title('SEEKER YAW ANGLE vs TIME, PROPNAV')
xlabel('TIME - SEC'),ylabel('RAD')
print -dps Xseeker2p1
!pstoepsi Xseeker2p1.ps Xseeker2p1.epsi
pause,clg
```

```
plot(time,Xseeker(2,:)),title('SEEKER PITCH ANGLE RATE vs TIME,
                              PROPNAV')
xlabel('TIME - SEC'),ylabel('RAD/SEC')
print -dps Xseeker3p1
!pstoepsi Xseeker3p1.ps Xseeker3p1.epsi
pause,clg
plot(time,Xseeker(4,:)),title('SEEKER YAW ANGLE RATE vs TIME,
                              PROPNAV')
xlabel('TIME - SEC'),ylabel('RAD/SEC')
print -dps Xseeker4p1
!pstoepsi Xseeker4p1.ps Xseeker4p1.epsi
pause,clg
% Range information
plot(time,r(4,:)),title('RANGE vs TIME, PROPNAV')
xlabel('TIME - SEC'),ylabel('FEET')
print -dps rp1
!pstoepsi rp1.ps rp1.epsi
pause,clg
% Missile velocity information
plot(time,vm),title('MISSILE SPEED vs TIME, PROPNAV')
xlabel('TIME - SEC'),ylabel('FEET/SEC')
print -dps vmp1
!pstoepsi vmp1.ps vmp1.epsi
pause,clg
% Target velocity information
plot(time,vt),title('TARGET SPEED vs TIME, PROPNAV')
xlabel('TIME - SEC'),ylabel('FEET/SEC')
print -dps vtp1
```

```
!pstoepsi vtp1.ps vtp1.epsi
pause,clg
% Closing velocity information
plot(time,vc_pitch),title('PITCH CLOSING SPEED, PROPNAV')
xlabel('TIME - SEC'),ylabel('FEET/SEC')
print -dps vc1p1
!pstoepsi vc1p1.ps vc1p1.epsi
pause,clg
plot(time,vc_yaw),title('YAW CLOSING SPEED vs TIME, PROPNAV')
xlabel('TIME - SEC'),ylabel('FEET/SEC')
print -dps vc2p1
!pstoepsi vc2p1.ps vc2p1.epsi
pause,clg
```

APPENDIX B - MISSILE/TARGET THREE DIMENSIONAL SIMULATION USING AUGMENTED PROPORTIONAL NAVIGATION GUIDANCE

```
% Written by: Rui Manuel Alves Francisco
% Date: 10 November 1992
% This Program simulates the terminal phase of a 3-D missile/target
% engagement using augmented proportional navigation guidance.

clear
clg
% DEFINE CONSTANTS
dt = .01; % Sampling time
Tf = 100; % maximum simulation time
kmax = Tf/dt+1;
n = 3; % navigation constant
N = [n 0
     0 n];
% DEFINE STATE EQUATIONS
% Missile
% Xm = [xm = missile's x coordinate
%       xmd = missile's speed (x coordinate)
%       ym = missile's y coordinate
%       ymd = missile's speed (y coordinate)
%       zm = missile's z coordinate
%       zmd = missile's speed (z coordinate)]
Am = [0 1 0 0 0 0
```

```
        0 0 0 0 0 0
        0 0 0 1 0 0
        0 0 0 0 0 0
        0 0 0 0 0 1
        0 0 0 0 0 0];
Bm = [0 0 0
        1 0 0
        0 0 0
        0 1 0
        0 0 0
        0 0 1];
% Target
% Xt = [xt = target's x coordinate
%         xtd = target's speed (x coordinate)
%          yt = target's y coordinate
%          ytd = target's speed (y coordinate)
%          zt = target's z coordinate
%          ztd = target's speed (z coordinate)]
At = [0 1 0 0 0 0
      0 0 0 0 0 0
      0 0 0 1 0 0
      0 0 0 0 0 0
      0 0 0 0 0 1
      0 0 0 0 0 0];
Bt = [0 0 0
      1 0 0
      0 0 0
      0 1 0
```

```
        0 0 0
        0 0 1];
% Seeker (Radar)
% Xsk = [beta_pitch = seeker's pitch angle
%        betad_pitch = seeker's pitch angle rate
%        beta_yaw = seeker's yaw angle
%        betad_yaw = seeker's yaw angle rate]
Ask = [ 0     1      0    0
      -100  -20      0    0
         0    0      0    1
         0    0   -100  -20];
Bsk = [ 0 0
        100   0
          0   0
          0 100];
% Guidance System
% Xgs = [a_m_pitch = missile's pitch acceleration
%        a_m_yaw = missile's yaw acceleration]
Ags = [-1 0
        0 -1];
Bgs = [1 0
       0 1];
% INITIALIZE STATE VARIABLES (when the missile enters into the terminal
 phase of flight)
% Missile
Xm(:,1) = [    0              % The missile is in a collision triangle
            2828              % with the target when the missile enters into
               0              % the terminal phase of flight
```

```
            1000
               0
            47.1339];
% Target
Xt(:,1) = [30000
      0
      0
   1000
    500
      0];
% DISCRETE REPRESENTATION
[PHIm,DELm] = c2d(Am,Bm,dt);
[PHIt,DELt] =  c2d(At,Bt,dt);
[PHIsk,DELsk] = c2d(Ask,Bsk,dt);
[PHIgs,DELgs] = c2d(Ags,Bgs,dt);
% LINE OF SIGHT (LOS) INFORMATION. INITIAL CONDITIONS.
% Missile
% LAMBDA_m = Missile's LOS from the global coordinate system
% LAMBDA_m = [LAMBDA_m_pitch = Missile's pitch LOS angle
%                      LAMBDA_m_yaw = Missile's yaw LOS angle]
LAMBDA_m(:,1) = [atan2(Xm(5,1),sqrt(Xm(1,1)^2+Xm(3,1)^2));
                      atan2(Xm(3,1),Xm(1,1))];
% Target
% LAMBDA_t = Target's LOS from the global coordinate system
% LAMBDA_t = [LAMBDA_t_pitch = Target's pitch LOS angle
%                      LAMBDA_t_yaw = Target's yaw LOS angle]
LAMBDA_t(:,1) = [atan2(Xt(5,1),sqrt(Xt(1,1)^2+Xt(3,1)^2));
                      atan2(Xt(3,1),Xt(1,1))];
```

```
% LOS from Missile to Target
% LAMBDA = LOS from Missile to Target.
% LAMBDA = [LAMBDA_pitch = LOS angle in pitch
%                   LAMBDA_yaw = LOS angle in yaw];
LAMBDA(:,1) = [atan2((Xt(5,1)-Xm(5,1)),sqrt((Xt(1,1)-Xm(1,1))^2
                    +(Xt(3,1)-Xm(3,1))^2));
                    atan2((Xt(3,1)-Xm(3,1)),abs(Xt(1,1)-Xm(1,1)))];
% MISSILE and TARGET FLIGHT PATH ANGLES INFORMATION
% Missile
%GAMMA_m = [GAMMA_m_pitch = Missile's flight path angle in pitch
%                    GAMMA_m_yaw = Missile's flight path angle in yaw]
GAMMA_m(:,1) = [atan2(Xm(6,1),sqrt(Xm(2,1)^2+Xm(4,1)^2));
                      atan2(Xm(4,1),Xm(2,1))];
% Target
%GAMMA_t = [GAMMA_t_pitch = Target's flight path angle in pitch
%                    GAMMA_t_YAW = Target's flight path angle in yaw]
GAMMA_t(:,1) = [atan2(Xt(6,1),sqrt(Xt(2,1)^2+Xt(4,1)^2));
                      atan2(Xt(4,1),Xt(2,1))];
% RANGE INFORMATION
% Missile
% Rm = Missile's range
Rm(1) = sqrt(Xm(1,1)^2 + Xm(3,1)^2 + Xm(5,1)^2);
% Target
% Rt = Target's range
Rt(1) = sqrt(Xt(1,1)^2 + Xt(3,1)^2 + Xt(5,1)^2);
% Missile/Target relative distance
% R = [Rmtx = Missile/Target x coordinate range
%       Rmty = Missile/Target y coordinate range
```

```
%        Rmtz = Missile/Target z coordinate range
% Rmt = Missile/Target relative distance(Rmt=sqrt(Rmtx^2+Rmty^2+Rmtz^2))]
R(:,1) = [Xt(1,1)-Xm(1,1)
          Xt(3,1)-Xm(3,1)
          Xt(5,1)-Xm(5,1)
    sqrt((Xt(1,1)-Xm(1,1))^2+(Xt(3,1)-Xm(3,1))^2+(Xt(5,1)-Xm(5,1))^2)];
% VELOCITY INFORMATION
% Missile
% Vm = Missile's Speed
Vm(1) = sqrt(Xm(2,1)^2+Xm(4,1)^2+Xm(6,1)^2);
% Target
% Vt = Target's Speed
Vt(1) = sqrt(Xt(2,1)^2+Xt(4,1)^2+Xt(6,1)^2);
% Speed along the pitch and yaw LOS. Pitch and yaw closing speeds
Vt_pitch(1) = Vt(1)*cos(LAMBDA(2,1)-GAMMA_t(2,1));
Vm_pitch(1) = Vm(1)*cos(LAMBDA(2,1)-GAMMA_m(2,1));
Vc_pitch(1) = Vm_pitch(1)*cos(GAMMA_m(1,1)-LAMBDA(1,1))
                -Vt_pitch(1)*cos(GAMMA_t(1,1)-LAMBDA(1,1));
Vt_yaw(1) = Vt(1)*cos(GAMMA_t(1,1));
Vm_yaw(1) = Vm(1)*cos(GAMMA_m(1,1));
Vc_yaw(1) = Vm_yaw(1)*cos(GAMMA_m(2,1)-LAMBDA(2,1))
              -Vt_yaw(1)*cos(GAMMA_t(2,1)-LAMBDA(2,1));
Vc = [Vc_pitch(1) 0
      0           Vc_yaw(1)];
% SEEKER and GUIDANCE SYSTEM INITIAL CONDITIONS and INPUTS.
% Seeker
Xsk(:,1) = [0
            0
```

```
                0
                0];
Usk(:,1) = LAMBDA(:,1); % Seeker input
% Guidance System
Xgs(:,1) = [0
                0];
Ugs(:,1) = N*Vc*[Xsk(2,1) % Guidance system input
                        Xsk(4,1)];
% Initial conditions of the target's acceleration
 a_tx(1)          = 0;
a_ty(1)           = 0;
a_tz(1)           = 0;
a_t(1)            = 0;
a_t_ pitch(1)     = 0;
a_t_yaw(1)        = 0;
TETA_t(1)          = 0; % angle between the acceleration vector and the yaw plane
PHI_t(1)           = 0; % yaw angle of the target's acceleration vector
% TIME
% Time = Time vector
TIME(1) = 0;
% Tgo = Time to go
Tgo(1) = R(4,1)/Vc_pitch(1);
% SIMULATE THE SYSTEM
for ti = 0:.25:21.25
 for i = 1:kmax-1
% Calculate components of the missile's pitch acceleration vector
a_mx_pitch(i) = -(Xgs(1,i)*sin(LAMBDA(1,i))*cos(LAMBDA(2,i)));
a_my_pitch(i) = -(Xgs(1,i)*sin(LAMBDA(1,i))*sin(LAMBDA(2,i)));
```

```
a_mz_pitch(i) = Xgs(1,i)*cos(LAMBDA(1,i));
% Calculate components of the missile's yaw acceleration vector
a_mx_yaw(i) = -Xgs(2,i)*sin(LAMBDA(2,i));
a_my_yaw(i) = Xgs(2,i)*cos(LAMBDA(2,i));
% Compute overall missile acceleration components
a_mx(i) = a_mx_pitch(i)+a_mx_yaw(i);
a_my(i) = a_my_pitch(i)+a_my_yaw(i);
a_mz(i) = a_mz_pitch(i);
am(:,i) = [a_mx(i)
           a_my(i)
           a_mz(i)];
% target acceleration vector
at(:,i) = [a_tx(i)
           a_ty(i)
           a_tz(i)];
% Compute magnitude of the missile's acceleration
a_m(i) = sqrt(a_mx(i)^2 + a_my(i)^2 + a_mz(i)^2);
% Generate target's evasive maneuver (we assume that these accelerations, along
% the three cartesian axis, are estimated using the missile's image processing
% capabilities)
if TIME(i) >= ti/2 % target starts evasive maneuver
  a_tx(i+1) = 3*32.2*sin(GAMMA_t(2,i));
  a_ty(i+1) = 4*32.2*cos(GAMMA_t(2,i));
  a_tz(i+1) = 3*32.2*cos(GAMMA_t(1,i));
else
  a_tx(i+1) = 0.0;
  a_ty(i+1) = 0.0;
  a_tz(i+1) = 0.0;
```

```
end
% Compute magnitude of the target's acceleration
a_t(i+1) = sqrt(a_tx(i+1)^2 + a_ty(i+1)^2 + a_tz(i+1)^2);
% Update missile states
Xm(:,i+1) = PHIm*Xm(:,i)+DELm*am(:,i);
% Update target states
Xt(:,i+1) = PHIt*Xt(:,i)+DELt*at(:,i);
% Update flight path angles
GAMMA_m(:,i+1) = [atan2(Xm(6,i+1),sqrt(Xm(2,i+1)^2+Xm(4,i+1)^2));
                  atan2(Xm(4,i+1),Xm(2,i+1))];
GAMMA_t(:,i+1) = [atan2(Xt(6,i+1),sqrt(Xt(2,i+1)^2+Xt(4,i+1)^2));
                  atan2(Xt(4,i+1),Xt(2,i+1))];
% Update seeker states
Xsk(:,i+1) = PHIsk*Xsk(:,i)+DELsk*Usk(:,i);
% Update Guidance System states
Xgs(:,i+1) = PHIgs*Xgs(:,i)+DELgs*Ugs(:,i);
% Limit yaw and pitch accelerations to 25 g's
 if abs(Xgs(1,i+1)) > 805.0
    Xgs(1,i+1) = 805.0 *sign(Xgs(1,i+1));
 end
 if abs(Xgs(2,i+1)) > 805.0
    Xgs(2,i+1) = 805.0 *sign(Xgs(2,i+1));
 end
% Update LOS angles
LAMBDA(:,i+1) = [atan2((Xt(5,i+1)-Xm(5,i+1)),sqrt((Xm(1,i+1)-Xt(1,i+1))^2
                  +(Xm(3,i+1)-Xt(3,i+1))^2));
                 atan2((Xt(3,i+1)-Xm(3,i+1)),(abs(Xm(1,i+1)-Xt(1,i+1))))];
Usk(:,i+1) = LAMBDA(:,i+1);
```

```
LAMBDA_m(:,i+1) = [atan2(Xm(5,i+1),sqrt(Xm(1,i+1)^2+Xm(3,i+1)^2));
                   atan2(Xm(3,i+1),Xm(1,i+1))];
LAMBDA_t(:,i+1) = [atan2(Xt(5,i+1),sqrt(Xt(1,i+1)^2+Xt(3,i+1)^2));
                   atan2(Xt(3,i+1),Xt(1,i+1))];
% Update Range Information
Rm(i+1) = sqrt(Xm(1,i+1)^2 + Xm(3,i+1)^2 + Xm(5,i+1)^2);
Rt(i+1) = sqrt(Xt(1,i+1)^2 + Xt(3,i+1)^2 + Xt(5,i+1)^2);
R(:,i+1) = [Xt(1,i+1)-Xm(1,i+1);
            Xt(3,i+1)-Xm(3,i+1);
            Xt(5,i+1)-Xm(5,i+1);
            sqrt((Xt(1,i+1)-Xm(1,i+1))^2+(Xt(3,i+1)-Xm(3,i+1))^2
            +(Xt(5,i+1)-Xm(5,i+1))^2)];
% Update Velocity Information
Vm(i+1) = sqrt(Xm(2,i+1)^2+Xm(4,i+1)^2+Xm(6,i+1)^2);
Vt(i+1) = sqrt(Xt(2,i+1)^2+Xt(4,i+1)^2+Xt(6,i+1)^2);
Vt_pitch(i+1) = Vt(i+1)*cos(LAMBDA(2,i+1)-GAMMA_t(2,i+1));
Vm_pitch(i+1) = Vm(i+1)*cos(LAMBDA(2,i+1)-GAMMA_m(2,i+1));
Vc_pitch(i+1) = Vm_pitch(i+1)*cos(GAMMA_m(1,i+1)-LAMBDA(1,i+1))
                -Vt_pitch(i+1)*cos(GAMMA_t(1,i+1)-LAMBDA(1,i+1));
Vt_yaw(i+1) = Vt(i+1)*cos(GAMMA_t(1,i+1));
Vm_yaw(i+1) = Vm(i+1)*cos(GAMMA_m(1,i+1));
Vc_yaw(i+1) = Vm_yaw(i+1)*cos(GAMMA_m(2,i+1)-LAMBDA(2,i+1))
                -Vt_yaw(i+1)*cos(GAMMA_t(2,i+1)-LAMBDA(2,i+1));
Vc = [Vc_pitch(i+1) 0
      0             Vc_yaw(i+1)];
% Calculate angles of the target's acceleration
TETA_t(i+1) = atan2(a_tz(i+1),sqrt(a_tx(i+1)^2+a_ty(i+1)^2));
PHI_t(i+1) = atan2(a_ty(i+1),a_tx(i+1));
```

```
% Calculate the components of the target's acceleration normal to the LOS
a_t_pitch(i+1) = -a_t(i+1)*cos(LAMBDA(2,i+1)-PHI_t(i+1))
                    *sin(LAMBDA(1,i+1)-TETA_t(i+1));
a_t_yaw(i+1) = -a_t(i+1)*cos(TETA_t(i+1))*sin(LAMBDA(2,i+1)-PHI_t(i+1));
% Update guidance system input
Ugs(:,i+1) = N*(Vc*[Xsk(2,i+1);Xsk(4,i+1)]
                    +.5*[a_t_pitch(i+1);a_t_yaw(i+1)]);
% Update Time/time to go
TIME(i+1) = TIME(i)+dt;
Tgo(i+1) = R(4,i+1)/Vc_pitch(i+1);
% Check for closest point
if (R(4,i) < R(4,i+1)),break,end
end;
% Save information for plotting and evaluation
R1(4*ti+1) = R(4,i); % miss distance
Ti(4*ti+1) = ti/2;% starting time of evasive maneuver (EM)
tgo(4*ti+1) = i*dt-ti/2; % time to go until end of flight
 if ti == 12.0          % Record information for a target that
                        % initialized the evasive maneuver 6 sec
                        % after the missile entered  into the terminal
  TGO = tgo(49);    % phase of flight
  Xseeker = Xsk(:,1:i);
  Xgsys = Xgs(:,1:i);
  lambda_m = LAMBDA_m(:,1:i);
  lambda_t = LAMBDA_t(:,1:i);
  lambda = LAMBDA(:,1:i);
  gamma_m = GAMMA_m(:,1:i);
  gamma_t = GAMMA_t(:,1:i);
```

```
  r = R(:,1:i);
  vm = Vm(1:i);
  vt = Vt(1:i);
  vm_pitch = Vm_pitch(1:i);
  vt_pitch = Vt_pitch(1:i);
  vm_yaw = Vm_yaw(1:i);
  vt_yaw = Vt_yaw(1:i);
  vc_pitch = Vc_pitch(1:i);
  vc_yaw = Vc_yaw(1:i);
  tGO = Tgo(1:i);
  a_M = am(:,1:i);
  a_T = at(:,1:i);
  A_t = a_t(1:i);
  A_m = a_m(1:i);
  time = TIME(1:i);
 end
clear R;
R(:,1) = [Xt(1,1)-Xm(1,1);
          Xt(3,1)-Xm(3,1);
          Xt(5,1)-Xm(5,1);
    sqrt((Xt(1,1)-Xm(1,1))^2+(Xt(3,1)-Xm(3,1))^2+(Xt(5,1)-Xm(5,1))^2)];
end;
save thesis2a343 R1 tgo Ti tGO missile target TGO Xseeker Xgsys lambda_m
 lambda_t lambda gamma_m gamma_t r vm vt vm_pitch vm_yaw vt_pitch
vt_yaw vc_pitch vc_yaw a_M a_T time A_t A_m
PLOTS
% Miss distance information
plot(Ti,R1),title('MISS DISTANCE vs INITIAL TIME, APROPNAV')
```

```
xlabel('INITIAL TIME - SEC'),ylabel('MISS - FEET')
print -dps R1aa1
!pstoepsi R1aa1.ps R1aa1.epsi
pause,clg
plot(tgo,R1),title('MISS DISTANCE vs TIME TO GO, APROPNAV')
xlabel('TIME TO GO - SEC'),ylabel('MISS - FEET')
print -dps R1ba1
!pstoepsi R1ba1.ps R1ba1.epsi
pause,clg
% Missile acceleration information
plot(time,A_m),title('MISSILE ACCELERATION MAGNITUDE vs TIME,
                      APROPNAV')
xlabel('TIME - SEC'),ylabel('FEET/SEC^2')
print -dps A_ma1
!pstoepsi A_ma1.ps A_ma1.epsi
pause,clg
plot(time,Xgsys(1,:)),title('MISSILE PITCH ACCELERATION vs TIME,
                             APROPNAV')
xlabel('TIME - SEC'),ylabel('FEET/SEC^2')
print -dps Xgsys1a1
!pstoepsi Xgsys1a1.ps Xgsys1a1.epsi
pause,clg
plot(time,Xgsys(2,:)),title('MISSILE YAW ACCELERATION vs TIME,
                             APROPNAV')
xlabel('TIME - SEC'),ylabel('FEET/SEC^2')
print -dps Xgsys2a1
!pstoepsi Xgsys2a1.ps Xgsys2a1.epsi
pause,clg
```

```
% Target acceleration information
plot(time,A_t),title('TARGET ACCELERATION MAGNITUDE vs TIME,
                    APROPNAV')
xlabel('TIME - SEC'),ylabel('FEET/SEC^2')
print -dps A_ta1
!pstoepsi A_ta1.ps A_ta1.epsi
pause,clg
% Seeker pitch and yaw angles
plot(time,Xseeker(1,:)),title('SEEKEK PITCH ANGLE vs TIME,APROPNAV')
xlabel('TIME - SEC'),ylabel('RAD')
print -dps Xseeker1a1
!pstoepsi Xseeker1a1.ps Xseeker1a1.epsi
pause,clg
plot(time,Xseeker(3,:)),title('SEEKER YAW ANGLE vs TIME, APROPNAV')
xlabel('TIME - SEC'),ylabel('RAD')
print -dps Xseeker2a1
!pstoepsi Xseeker2a1.ps Xseeker2a1.epsi
pause,clg
plot(time,Xseeker(2,:)),title('SEEKER PITCH ANGLE RATE vs TIME,
                    APROPNAV')
xlabel('TIME - SEC'),ylabel('RAD/SEC')
print -dps Xseeker3a1
!pstoepsi Xseeker3a1.ps Xseeker3a1.epsi
pause,clg
plot(time,Xseeker(4,:)),title('SEEKER YAW ANGLE RATE vs TIME,
                    APROPNAV')
xlabel('TIME - SEC'),ylabel('RAD/SEC')
print -dps Xseeker4a1
```

```
!pstoepsi Xseeker4a1.ps Xseeker4a1.epsi
pause,clg
% Range information
plot(time,r(4,:)),title('RANGE vs TIME, APROPNAV')
xlabel('TIME - SEC'),ylabel('FEET')
print -dps ra1
!pstoepsi ra1.ps ra1.easi
pause,clg
% Missile velocity information
plot(time,vm),title('MISSILE SPEED vs TIME, APROPNAV')
xlabel('TIME - SEC'),ylabel('FEET/SEC')
print -dps vma1
!pstoepsi vma1.ps vma1.epsi
pause,clg
% Target velocity information
plot(time,vt),title('TARGET SPEED vs TIME, APROPNAV')
xlabel('TIME - SEC'),ylabel('FEET/SEC')
print -dps vta1
!pstoepsi vta1.ps vta1.epsi
pause,clg
% Closing velocity information
plot(time,vc_pitch),title('PITCH CLOSING SPEED,APROPNAV')
xlabel('TIME - SEC'),ylabel('FEET/SEC')
print -dps vc1a1
!pstoepsi vc1a1.ps vc1a1.epsi
pause,clg
plot(time,vc_yaw),title('YAW CLOSING SPEED vs TIME, APROPNAV')
xlabel('TIME - SEC'),ylabel('FEET/SEC')
```

```
print -dps vc2a1
!pstoepsi vc2a1.ps vc2a1.epsi
pause,clg
```

APPENDIX C - MISSILE/TARGET THREE DIMENSIONAL SIMULATION USING OPTIMAL GUIDANCE

```
% Written by: Rui Manuel Alves Francisco
% Date: 09 December 1992
% This Program simulates the terminal phase of a 3-D missile/target
% engagement using optimal guidance.

clear
clg
% DEFINE CONSTANTS
dt = .01; % Sampling time
Tf = 100; % maximum simulation time
kmax = Tf/dt+1;
% DEFINE STATE EQUATIONS
% Missile
% Xm = [xm = missile's x coordinate
%          xmd = missile's speed (x coordinate)
%           ym = missile's y coordinate
%           ymd = missile's speed (y coordinate)
%           zm = missile's z coordinate
%       zmd = missile's speed (z coordinate)]
Am = [0 1 0 0 0 0
      0 0 0 0 0 0
      0 0 0 1 0 0
      0 0 0 0 0 0
      0 0 0 0 0 1
```

```
        0 0 0 0 0 0];
Bm = [0 0 0
        1 0 0
        0 0 0
        0 1 0
        0 0 0
        0 0 1];
% Target

% Xt = [xt = target's x coordinate
%         xtd = target's speed (x coordinate)
%          yt = target's y coordinate
%          ytd = target's speed (y coordinate)
%          zt = target's z coordinate
%          ztd = target's speed (z coordinate)]
At = [0 1 0 0 0 0
      0 0 0 0 0 0
      0 0 0 1 0 0
      0 0 0 0 0 0
      0 0 0 0 0 1
      0 0 0 0 0 0];
Bt = [0 0 0
      1 0 0
      0 0 0
      0 1 0
      0 0 0
      0 0 1];
% Seeker (Radar)
```

```
% Xsk = [beta_pitch = seeker's pitch angle
%        betad_pitch = seeker's pitch angle rate
%        beta_yaw = seeker's yaw angle
%        betad_yaw = seeker's yaw angle rate]
Ask = [ 0    1      0    0
      -100  -20     0    0
        0    0      0    1
        0    0   -100  -20];
Bsk = [ 0    0
      100    0
        0    0
        0  100];
% Guidance System
% Xgs = [a_m_pitch = missile's pitch acceleration
%        a_m_yaw = missile's yaw acceleration]
Ags = [-1 0
        0 -1];
Bgs = [1 0
       0 1];
% INITIALIZE STATE VARIABLES (when the missile enters into the terminal
 phase of flight)

% Missile
Xm(:,1) = [    0          % The missile is in a collision triangle
            2828          % with the target when the missile enters into
               0          % the terminal phase of flight
            1000
               0
```

```
                47.1339];
% Target
Xt(:,1) = [30000
        0
        0
    1000
     500
        0];
% DISCRETE REPRESENTATION
[PHIm,DELm] = c2d(Am,Bm,dt);
[PHIt,DELt] =  c2d(At,Bt,dt);
[PHIsk,DELsk] = c2d(Ask,Bsk,dt);
[PHIgs,DELgs] = c2d(Ags,Bgs,dt);
% LINE OF SIGHT (LOS) INFORMATION. INITIAL CONDITIONS.
% Missile
% LAMBDA_m = Missile's LOS from the global coordinate system
% LAMBDA_m = [LAMBDA_m_pitch = Missile's pitch LOS angle
%                       LAMBDA_m_yaw = Missile's yaw LOS angle]
LAMBDA_m(:,1) = [atan2(Xm(5,1),sqrt(Xm(1,1)^2+Xm(3,1)^2));
                        atan2(Xm(3,1),Xm(1,1))];
% Target
% LAMBDA_t = Target's LOS from the global coordinate system
% LAMBDA_t = [LAMBDA_t_pitch = Target's pitch LOS angle
%                       LAMBDA_t_yaw = Target's yaw LOS angle]
LAMBDA_t(:,1) = [atan2(Xt(5,1),sqrt(Xt(1,1)^2+Xt(3,1)^2));
                        atan2(Xt(3,1),Xt(1,1))];
% LOS from Missile to Target
% LAMBDA = LOS from Missile to Target.
```

```
% LAMBDA = [LAMBDA_pitch = LOS angle in pitch
%                  LAMBDA_yaw = LOS angle in yaw];
LAMBDA(:,1) = [atan2((Xt(5,1)-Xm(5,1)),sqrt((Xt(1,1)-Xm(1,1))^2
                    +(Xt(3,1)-Xm(3,1))^2));
                    atan2((Xt(3,1)-Xm(3,1)),abs(Xt(1,1)-Xm(1,1)))];
% MISSILE and TARGET FLIGHT PATH ANGLES INFORMATION
% Missile
%GAMMA_m = [GAMMA_m_pitch = Missile's flight path angle in pitch
%                  GAMMA_m_yaw = Missile's flight path angle in yaw]
GAMMA_m(:,1) = [atan2(Xm(6,1),sqrt(Xm(2,1)^2+Xm(4,1)^2));
                      atan2(Xm(4,1),Xm(2,1))];
% Target
%GAMMA_t = [GAMMA_t_pitch = Target's flight path angle in pitch
%                  GAMMA_t_YAW = Target's flight path angle in yaw]
GAMMA_t(:,1) = [atan2(Xt(6,1),sqrt(Xt(2,1)^2+Xt(4,1)^2));
                      atan2(Xt(4,1),Xt(2,1))];
% RANGE INFORMATION
% Missile
% Rm = Missile's range
Rm(1) = sqrt(Xm(1,1)^2 + Xm(3,1)^2 + Xm(5,1)^2);
% Target
% Rt = Target's range
Rt(1) = sqrt(Xt(1,1)^2 + Xt(3,1)^2 + Xt(5,1)^2);
% Missile/Target relative distance
% R = [Rmtx = Missile/Target x coordinate range
%      Rmty = Missile/Target y coordinate range
%      Rmtz = Missile/Target z coordinate range
% Rmt = Missile/Target relative distance(Rmt=sqrt(Rmtx^2+Rmty^2+Rmtz^2))]
```

```
R(:,1) = [Xt(1,1)-Xm(1,1)
          Xt(3,1)-Xm(3,1)
          Xt(5,1)-Xm(5,1)
    sqrt((Xt(1,1)-Xm(1,1))^2+(Xt(3,1)-Xm(3,1))^2+(Xt(5,1)-Xm(5,1))^2)];
% VELOCITY INFORMATION
% Missile
% Vm = Missile's Speed
Vm(1) = sqrt(Xm(2,1)^2+Xm(4,1)^2+Xm(6,1)^2);
% Target
% Vt = Target's Speed
Vt(1) = sqrt(Xt(2,1)^2+Xt(4,1)^2+Xt(6,1)^2);
% Speed along the pitch and yaw LOS. Pitch and yaw closing speeds
Vt_pitch(1) = Vt(1)*cos(LAMBDA(2,1)-GAMMA_t(2,1));
Vm_pitch(1) = Vm(1)*cos(LAMBDA(2,1)-GAMMA_m(2,1));
Vc_pitch(1) = Vm_pitch(1)*cos(GAMMA_m(1,1)-LAMBDA(1,1))
                -Vt_pitch(1)*cos(GAMMA_t(1,1)-LAMBDA(1,1));
Vt_yaw(1) = Vt(1)*cos(GAMMA_t(1,1));
Vm_yaw(1) = Vm(1)*cos(GAMMA_m(1,1));
Vc_yaw(1) = Vm_yaw(1)*cos(GAMMA_m(2,1)-LAMBDA(2,1))
              -Vt_yaw(1)*cos(GAMMA_t(2,1)-LAMBDA(2,1));
Vc = [Vc_pitch(1) 0
      0           Vc_yaw(1)];
% Tgo = Time to go
Tgo(1) = R(4,1)/Vc_pitch(1);
% Optimal guidance coefficients
k= Tgo(1);
n = (6*k^2*(exp(-k)-1+k))/(2*k^3+3+6*k-6*k^2-12*k*exp(-k)-3*exp(-2*k));
N = [n  0
```

```
      0  n];
% SEEKER and GUIDANCE SYSTEM INITIAL CONDITIONS and INPUTS.
% Seeker
Xsk(:,1) = [0
            0
            0
            0];
Usk(:,1) = LAMBDA(:,1); % Seeker input
% Guidance System
Xgs(:,1) = [0
            0];
Ugs(:,1) = N*Vc*[Xsk(2,1) % Guidance system input
                 Xsk(4,1)];
% Initial conditions of the target's acceleration
 a_tx(1)          = 0;
a_ty(1)           = 0;
a_tz(1)           = 0;
a_t(1)            = 0;
a_ t_pitch(1)     = 0;
a_t_yaw(1)        = 0;
TETA_t(1)         = 0; % angle between the acceleration vector and the yaw plane
PHI_t(1)          = 0; % yaw angle of the target's acceleration vector
% Initial conditions of the missile's acceleration
a_ m_pitch(1)         = 0;
a_m_yaw(1)            = 0;
% TIME
% Time = Time vector
TIME(1) = 0;
```

```
% SIMULATE THE SYSTEM
for ti = 0:.25:21.25
  for i = 1:kmax-1
% Calculate components of the missile's pitch acceleration vector
a_mx_pitch(i) = -(Xgs(1,i)*sin(LAMBDA(1,i))*cos(LAMBDA(2,i)));
a_my_pitch(i) = -(Xgs(1,i)*sin(LAMBDA(1,i))*sin(LAMBDA(2,i)));
a_mz_pitch(i) = Xgs(1,i)*cos(LAMBDA(1,i));
% Calculate missile's yaw acceleration vector components
a_mx_yaw(i) = -Xgs(2,i)*sin(LAMBDA(2,i));
a_my_yaw(i) = Xgs(2,i)*cos(LAMBDA(2,i));
% Compute overall missile acceleration
a_mx(i) = a_mx_pitch(i)+a_mx_yaw(i);
a_my(i) = a_my_pitch(i)+a_my_yaw(i);
a_mz(i) = a_mz_pitch(i);
am(:,i) = [a_mx(i)
           a_my(i)
           a_mz(i)];
% target acceleration vector
at(:,i) = [a_tx(i)
          a_ty(i)
          a_tz(i)];
% Compute magnitude of the missile's acceleration
a_m(i) = sqrt(a_mx(i)^2 + a_my(i)^2 + a_mz(i)^2);
% Generate target's evasive maneuver (we assume that these accelerations, along
% the three cartesian axis, are estimated using the missile's image processing
% capabilities)
if TIME(i) >= ti/2 % target starts evasive maneuver
  a_tx(i+1) = 3*32.2*sin(GAMMA_t(2,i));
```

```
  a_ty(i+1) = 4*32.2*cos(GAMMA_t(2,i));
  a_tz(i+1) = 3*32.2*cos(GAMMA_t(1,i));
else
  a_tx(i+1) = 0.0;
  a_ty(i+1) = 0.0;
  a_tz(i+1) = 0.0;
end
% Compute magnitude of the target's acceleration
a_t(i+1) = sqrt(a_tx(i+1)^2 + a_ty(i+1)^2 + a_tz(i+1)^2);
% Update missile states
Xm(:,i+1) = PHIm*Xm(:,i)+DELm*am(:,i);
% Update target states
Xt(:,i+1) = PHIt*Xt(:,i)+DELt*at(:,i);
% Update flight path angles
GAMMA_m(:,i+1) = [atan2(Xm(6,i+1),sqrt(Xm(2,i+1)^2+Xm(4,i+1)^2));
                  atan2(Xm(4,i+1),Xm(2,i+1))];
GAMMA_t(:,i+1) = [atan2(Xt(6,i+1),sqrt(Xt(2,i+1)^2+Xt(4,i+1)^2));
                  atan2(Xt(4,i+1),Xt(2,i+1))];
% Update seeker states
Xsk(:,i+1) = PHIsk*Xsk(:,i)+DELsk*Usk(:,i);
% Update Guidance System states
Xgs(:,i+1) = PHIgs*Xgs(:,i)+DELgs*Ugs(:,i);
% Limit yaw and pitch accelerations to 25 g's
 if abs(Xgs(1,i+1)) > 805.0
   Xgs(1,i+1) = 805.0 *sign(Xgs(1,i+1));
 end
 if abs(Xgs(2,i+1)) > 805.0
   Xgs(2,i+1) = 805.0 *sign(Xgs(2,i+1));
```

```
end
% Update LOS angles
LAMBDA(:,i+1) = [atan2((Xt(5,i+1)-Xm(5,i+1)),sqrt((Xm(1,i+1)-Xt(1,i+1))^2
                +(Xm(3,i+1)-Xt(3,i+1))^2));
               atan2((Xt(3,i+1)-Xm(3,i+1)),(abs(Xm(1,i+1)-Xt(1,i+1))))];
Usk(:,i+1) = LAMBDA(:,i+1);
LAMBDA_m(:,i+1) = [atan2(Xm(5,i+1),sqrt(Xm(1,i+1)^2+Xm(3,i+1)^2));
                   atan2(Xm(3,i+1),Xm(1,i+1))];
LAMBDA_t(:,i+1) = [atan2(Xt(5,i+1),sqrt(Xt(1,i+1)^2+Xt(3,i+1)^2));
                   atan2(Xt(3,i+1),Xt(1,i+1))];
% Update Range Information
Rm(i+1) = sqrt(Xm(1,i+1)^2 + Xm(3,i+1)^2 + Xm(5,i+1)^2);
Rt(i+1) = sqrt(Xt(1,i+1)^2 + Xt(3,i+1)^2 + Xt(5,i+1)^2);
R(:,i+1) = [Xt(1,i+1)-Xm(1,i+1);
            Xt(3,i+1)-Xm(3,i+1);
            Xt(5,i+1)-Xm(5,i+1);
            sqrt((Xt(1,i+1)-Xm(1,i+1))^2+(Xt(3,i+1)-Xm(3,i+1))^2
            +(Xt(5,i+1)-Xm(5,i+1))^2)];
% Update Velocity Information
Vm(i+1) = sqrt(Xm(2,i+1)^2+Xm(4,i+1)^2+Xm(6,i+1)^2);
Vt(i+1) = sqrt(Xt(2,i+1)^2+Xt(4,i+1)^2+Xt(6,i+1)^2);
Vt_pitch(i+1) = Vt(i+1)*cos(LAMBDA(2,i+1)-GAMMA_t(2,i+1));
Vm_pitch(i+1) = Vm(i+1)*cos(LAMBDA(2,i+1)-GAMMA_m(2,i+1));
Vc_pitch(i+1) = Vm_pitch(i+1)*cos(GAMMA_m(1,i+1)-LAMBDA(1,i+1))
               -Vt_pitch(i+1)*cos(GAMMA_t(1,i+1)-LAMBDA(1,i+1));
Vt_yaw(i+1) = Vt(i+1)*cos(GAMMA_t(1,i+1));
Vm_yaw(i+1) = Vm(i+1)*cos(GAMMA_m(1,i+1));
Vc_yaw(i+1) = Vm_yaw(i+1)*cos(GAMMA_m(2,i+1)-LAMBDA(2,i+1))
```

```
                -Vt_yaw(i+1)*cos(GAMMA_t(2,i+1)-LAMBDA(2,i+1));
Vc = [Vc_pitch(i+1) 0
      0             Vc_yaw(i+1)];
% Time to go
Tgo(i+1) = R(4,i+1)/Vc_pitch(i+1);
% Calculate angles of the target's acceleration
TETA_t(i+1) = atan2(a_tz(i+1),sqrt(a_tx(i+1)^2+a_ty(i+1)^2));
PHI_t(i+1) = atan2(a_ty(i+1),a_tx(i+1));
% Calculate the components of the target's acceleration normal to the LOS
a_t_pitch(i+1) = -a_t(i+1)*cos(LAMBDA(2,i+1)-PHI_t(i+1))
                 *sin(LAMBDA(1,i+1)-TETA_t(i+1));
a_t_yaw(i+1) = -a_t(i+1)*cos(TETA_t(i+1))*sin(LAMBDA(2,i+1)-PHI_t(i+1));
% Update optimal guidance coefficients
k = Tgo(i+1);
n = (6*k^2*(exp(-k)-1+k))/(2*k^3+3+6*k-6*k^2-12*k*exp(-k)-3*exp(-2*k));
N = [n  0
     0  n];
% Components of the Missile's acceleration normal to the pitch and yaw LOS
a_m_pitch(i+1) = Xgs(1,i+1);
a_m_yaw(i+1) = Xgs(2,i+1);
% Update guidance system input
Ugs(:,i+1) = N*(Vc*[Xsk(2,i+1);Xsk(4,i+1)]
                +.5*[a_t_pitch(i+1);a_t_yaw(i+1)])
                 -(1/k^2)*(exp(-k)+k-1)*[a_m_pitch(i+1);a_m_yaw(i+1)]);
% Update Time
TIME(i+1) = TIME(i)+dt;
% Check for closest point
if (R(4,i) < R(4,i+1)),break,end
```

```
end;
% Save information for plotting and evaluation
R1(4*ti+1) = R(4,i); % miss distance
Ti(4*ti+1) = ti/2;% starting time of evasive maneuver (EM)
tgo(4*ti+1) = i*dt-ti/2; % time to go until end of flight
 if ti == 12.0          % Record information for a target that
                        % initialized the evasive maneuver 6 sec after
                        % the missile entered into the terminal phase
  TGO = tgo(49);    % of flight
  Xseeker = Xsk(:,1:i);
  Xgsys = Xgs(:,1:i);
  lambda_m = LAMBDA_m(:,1:i);
  lambda_t = LAMBDA_t(:,1:i);
  lambda = LAMBDA(:,1:i);
  gamma_m = GAMMA_m(:,1:i);
  gamma_t = GAMMA_t(:,1:i);
  r = R(:,1:i);
  vm = Vm(1:i);
  vt = Vt(1:i);
  vm_pitch = Vm_pitch(1:i);
  vt_pitch = Vt_pitch(1:i);
  vm_yaw = Vm_yaw(1:i);
  vt_yaw = Vt_yaw(1:i);
  vc_pitch = Vc_pitch(1:i);
  vc_yaw = Vc_yaw(1:i);
  tGO = Tgo(1:i);
  a_M = am(:,1:i);
  a_T = at(:,1:i);
```

```
  A_t = a_t(1:i);
  A_m = a_m(1:i);
  time = TIME(1:i);
 end
clear R;
R(:,1) = [Xt(1,1)-Xm(1,1);
          Xt(3,1)-Xm(3,1);
          Xt(5,1)-Xm(5,1);
    sqrt((Xt(1,1)-Xm(1,1))^2+(Xt(3,1)-Xm(3,1))^2+(Xt(5,1)-Xm(5,1))^2)];
end;
save thesis3o343 R1 tgo Ti tGO missile target TGO Xseeker Xgsys lambda_m
 lambda_t lambda gamma_m gamma_t r vm vt vm_pitch vm_yaw vt_pitch
vt_yaw vc_pitch vc_yaw a_M a_T time A_t A_m
PLOTS
% Miss distance information
plot(Ti,R1),title('MISS DISTANCE vs INITIAL TIME, OPTIMAL')
xlabel('INITIAL TIME - SEC'),ylabel('MISS - FEET')
print -dps R1ao1
!pstoepsi R1ao1.ps R1ao1.epsi
pause,clg
plot(tgo,R1),title('MISS DISTANCE vs TIME TO GO, OPTIMAL')
xlabel('TIME TO GO - SEC'),ylabel('MISS - FEET')
print -dps R1bo1
!pstoepsi R1bo1.ps R1bo1.epsi
pause,clg
% Missile acceleration information
plot(time,A_m),title('MISSILE ACCELERATION MAGNITUDE vs TIME,
                      OPTIMAL')
```

```
xlabel('TIME - SEC'),ylabel('FEET/SEC^2')
print -dps A_mo1
!pstoepsi A_mo1.ps A_mo1.epsi
pause,clg
plot(time,Xgsys(1,:)),title('MISSILE PITCH ACCELERATION vs TIME,
                                OPTIMAL')
xlabel('TIME - SEC'),ylabel('FEET/SEC^2')
print -dps Xgsys1o1
!pstoepsi Xgsys1o1.ps Xgsys1o1.epsi
pause,clg
plot(time,Xgsys(2,:)),title('MISSILE YAW ACCELERATION vs TIME,
                                OPTIMAL')
xlabel('TIME - SEC'),ylabel('FEET/SEC^2')
print -dps Xgsys2o1
!pstoepsi Xgsys2o1.ps Xgsys2o1.epsi
pause,clg
% Target acceleration information
plot(time,A_t),title('TARGET ACCELERATION MAGNITUDE vs TIME,
                            OPTIMAL')
xlabel('TIME - SEC'),ylabel('FEET/SEC^2')
print -dps A_to1
!pstoepsi A_to1.ps A_to1.epsi
pause,clg
% Seeker pitch and yaw angles
plot(time,Xseeker(1,:)),title('SEEKER PITCH ANGLE vs TIME, OPTIMAL')
xlabel('TIME - SEC'),ylabel('RAD')
print -dps Xseeker1o1
!pstoepsi Xseeker1o1.ps Xseeker1o1.epsi
```

```
pause,clg
plot(time,Xseeker(3,:)),title('SEEKER YAW ANGLE vs TIME, OPTIMAL')
xlabel('TIME - SEC'),ylabel('RAD')
print -dps Xseeker2o1
!pstoepsi Xseeker2o1.ps Xseeker2o1.epsi
pause,clg
plot(time,Xseeker(2,:)),title('SEEKER PITCH ANGLE RATE vs TIME,
                                OPTIMAL')
xlabel('TIME - SEC'),ylabel('RAD/SEC')
print -dps Xseeker3o1
!pstoepsi Xseeker3o1.ps Xseeker3o1.epsi
pause,clg
plot(time,Xseeker(4,:)),title('SEEKER YAW ANGLE RATE vs TIME,
                                OPTIMAL')
xlabel('TIME - SEC'),ylabel('RAD/SEC')
print -dps Xseeker4o1
!pstoepsi Xseeker4o1.ps Xseeker4o1.epsi
pause,clg
% Range information
plot(time,r(4,:)),title('RANGE vs TIME, OPTIMAL')
xlabel('TIME - SEC'),ylabel('FEET')
print -dps ro1
!pstoepsi ro1.ps ro1.easi
pause,clg
% Missile velocity information
plot(time,vm),title('MISSILE SPEED vs TIME, OPTIMAL')
xlabel('TIME - SEC'),ylabel('FEET/SEC')
print -dps vmo1
```

```
!pstoepsi vmo1.ps vmo1.epsi
pause,clg
% Target velocity information
plot(time,vt),title('TARGET SPEED vs TIME, OPTIMAL')
xlabel('TIME - SEC'),ylabel('FEET/SEC')
print -dps vto1
!pstoepsi vto1.ps vto1.epsi
pause,clg
% Closing velocity information
plot(time,vc_pitch),title('PITCH CLOSING SPEED, OPTIMAL')
xlabel('TIME - SEC'),ylabel('FEET/SEC')
print -dps vc1o1
!pstoepsi vc1o1.ps vc1o1.epsi
pause,clg
plot(time,vc_yaw),title('YAW CLOSING SPEED vs TIME, OPTIMAL')
xlabel('TIME - SEC'),ylabel('FEET/SEC')
print -dps vc2o1
!pstoepsi vc2o1.ps vc2o1.epsi
pause,clg
```

REFERENCES

1. Zarchan, P., *Tactical and Strategic Missile Guidance*, American Institute of Aeronautics and Astronautics, Inc., 1990.

2. Guelman, M., "The closed form solution of true proportional navigation," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 472 - 482, July 1976.

3. Schalkoff, R. J., *Digital Image Processing and Computer Vision*, Wiley, 1989.

4. Weng, J., Ahuja, N. and Huang, T., "Two - view matching," *Proceedings 2nd Int. Conf. Computer Vision*, Florida, pp. 64 - 73, December 1988.

5. Weng, J., Ahuja, N. and Huang, T., "Motion and Structure from Two Perspective Views: Algorithms, Error Analysis, and Error Estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. II, no. 5, May 1989.

6. Bryson, A.E., Ho, Yu - Chi, *Applied Optimal Control: Optimization, Estimation and Control*, Hemisphere Publishing Corporation, 1975.

INITIAL DISTRIBUTION LIST

Defense Technical Information Center Cameron Station Alexandria, VA 22304 - 6145	2
Dudley Knox Library Code 52 Naval Postgraduate School Monterey, CA 93943	2
Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943	1
Professor J.B.Burl, Code EC/Bl Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943	2
Professor R.Cristi, Code EC/Cx Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943	2
LT Rui M. A. Francisco Rua Luis de Camoes #8 1/Dto S.Pedro do Estoril 2765 Estoril Portugal	4
DSIT - 1/a Rep Administracao Central de Marinha Rua do Arsenal 1100 Lisboa Portugal	2